S^2I-dLLM: From Sparsity and Specialization to Adaptive Acceleration in Multimodal Diffusion LLMs

Anonymous authors
Paper under double-blind review

Abstract

Multimodal diffusion large language models (dLLMs) integrate tasks across modalities within a shared bidirectional Transformer and a masked diffusion objective, yet how the computational demands of different tasks are organized within the shared network and evolve during denoising remains insufficiently understood. In this work, we analyze the computational demands of multimodal dLLMs from two perspectives, network structure and token representation evolution, characterizing how different tasks sparsely depend on shared computation and how tokens with different roles differ in representational stability and update requirements. Building on this analysis, we propose S^2I-dLLM, a training-free, task-aware acceleration framework that adaptively allocates computation according to the demands of different tasks and how these demands change throughout denoising. S^2I-dLLM reduces network channels and selects layers to execute according to task requirements, while restoring necessary computation as denoising progresses to control the accumulation of approximation errors. At the token level, the framework uses representational stability to guide feature reuse and prediction states to adjust the timing of output refinement and commitment, thereby coordinating per-step computational cost with generation progress. Experimental results demonstrate that S^2I-dLLM effectively improves the inference efficiency of multimodal dLLMs while preserving task performance.

1 Introduction

In recent years, large language models (LLMs) have achieved remarkable breakthroughs in text generation and complex reasoning tasks. Beyond the autoregressive paradigm, masked diffusion language models (dLLMs) introduce diffusion-based denoising into language modeling, bridging diffusion models and autoregressive models, and are emerging as a compelling alternative owing to their bidirectional attention mechanism and potential for parallel decoding (Austin et al., 2021; Sahoo et al., 2024), with representative works including LLaDA (Nie et al., 2026) and Dream (Ye et al., 2025). Building on this foundation, multimodal diffusion foundation models represented by MMaDA (Yang et al., 2026; Swerdlow et al., 2025; Shi et al., 2025; Xin et al., 2025; Li et al., 2026a; Bie et al., 2026) further integrate image or text tasks into a single bidirectional Transformer under one shared masked diffusion training objective, thereby endowing a single model with general-purpose generative and understanding capabilities. Representative dLLM acceleration methods, such as confidence-guided parallel decoding (Wu et al., 2026) or feature-similarity-guided adaptive caching (Liu et al., 2025), among others, implicitly assume that the model has homogeneous computational demands across all tasks. This exposes a structural tension: although different inference tasks share the same backbone, bidirectional attention mechanism, representation space, and masked diffusion objective, their intrinsic computational profiles differ fundamentally.

This discrepancy reveals a crucial but largely unexplored gap in our understanding. **Does every task require the full computation of the shared backbone?** If not, how do their computational demands differ within the shared architecture? Answering these questions is essential for identifying task-dependent redundancy and designing efficient, adaptive inference for multimodal dLLMs.

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

To answer these questions, we conduct the first systematic dissection of the internal computation of a multimodal dLLM. Our empirical analysis reveals a striking finding: despite operating within the same shared architecture, different tasks are far from computationally homogeneous. We observe **Task Structural Sparsity** within the shared network: despite using the same model parameters, different tasks exhibit markedly different importance rankings of FFN channels (LeCun et al., 1989) (Figure 1(a)); meanwhile, layer-wise computational analysis shows that tasks differ in both their representation evolution and their reliance on layer-wise computation (Figure 1(b)). Together, these results reveal that computational differences across tasks manifest in both channel usage and layer-wise dependencies, and are influenced by denoising progress. Having uncovered this structural sparsity, we turn our attention to the tokens that carry information for different tasks. We find that the evolution of token representations exhibits **Token Functional Specialization**: although conditioning tokens are given, bidirectional attention still allows them to interact with the content being generated, so their hidden representations also change during denoising, but typically less than those of target tokens, remaining relatively stable; target tokens, in contrast, continue to evolve, with patterns of change varying across tasks and denoising stages. This indicates that the computational demands of tokens are closely tied to their roles, providing a basis for selective computation and feature reuse.

Building upon these findings, we introduce **S^{2}I-dLLM — Sparse and Specialized Inference for dLLMs**, a training-free acceleration framework for multimodal dLLMs. Our core principle is to align computational allocation with the task structural sparsity and token functional specialization that emerge within the model, while adapting it throughout denoising. To exploit task structural sparsity, we retain the computation essential to the current task through task-aware channel selection and adaptive layer-wise computation (Section 3.3), while mitigating approximation errors through phase-aware width recovery and on-demand recomputation (Section 3.5). To exploit token functional specialization, we organize token-level computation around two interrelated decisions, feature refresh and generation progression: we select which features to recompute based on representational stability and correlation, and determine when to refine and commit generated outputs based on prediction states (Section 3.4). The spatial structure, contextual dependencies, and prediction uncertainty of each task jointly shape these two decisions, allowing computational allocation to adapt to the generation requirements of different tasks.

In summary, our contributions are threefold:

- We systematically investigate the internal computational mechanisms of multimodal dLLMs, uncovering task structural sparsity and token functional specialization, and characterize how computational demands differ across tasks within a shared architecture and evolve throughout denoising.
- We propose S^{2}I-dLLM, a training-free, task-aware acceleration framework for multimodal dLLMs, integrating task-aware computational sparsification and dynamic token scheduling into a unified execution framework to enable adaptive acceleration aligned with the model's internal computational characteristics.
- We design a shared execution plan and a computation recovery mechanism to coordinate structural reduction and token scheduling within the framework, restore necessary computation based on denoising progress and generation states, and rebuild the relevant caches after computational path adjustments to mitigate errors arising from sustained approximation.

2 Related Work

2.1 Diffusion-based Large Language Models

Discrete diffusion models learn to reverse corruption over categorical states (Austin et al., 2021; Hoogeboom et al., 2021), with score-entropy objectives (Lou et al., 2024) and simplified masked modeling (Sahoo et al., 2024) improving language generation. LLaDA and Dream (Nie et al., 2026; Ye et al., 2025) scale this paradigm to billion-parameter language models with bidirectional context modeling and parallel generation (Gong et al., 2025; Zhu et al., 2025). Multimodal extensions (Li et al., 2026b; You et al., 2026; Yu et al., 2025) incorporate visual conditioning into dLLMs for image understanding. In parallel, unified discrete diffusion models (Swerdlow et al., 2025; Shi et al.,

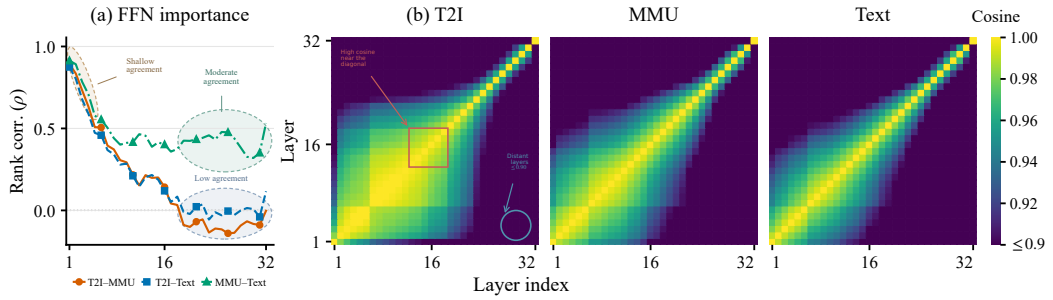


Figure 1: Computational characteristics of the shared network in MMaDA (Yang et al., 2026). (a) Cross-task correlations of FFN channel importance; lower correlations indicate greater differences in channel importance rankings across tasks. (b) Inter-layer cosine similarity of target-token representations, showing how representations evolve across network depth in different tasks.

2025) jointly model text and image generation. MMaDA (Yang et al., 2026) unifies text reasoning, multimodal understanding, and text-to-image generation under a shared masked diffusion framework, providing the backbone for our study of task-dependent computation. More recent unified models (Li et al., 2026a; Xin et al., 2025; Bie et al., 2026) further extend multimodal understanding and generation to tasks such as grounding, editing, and interleaved reasoning.

2.2 Efficient Inference for Diffusion Language Models

To accommodate bidirectional dependencies, approximate KV and feature caching reuse stable computation across denoising steps (Ma et al., 2026; Liu et al., 2025; Hu et al., 2026; Jiang et al., 2026). More selective schemes exploit token saliency for cache eviction (Song et al., 2026; Huang et al., 2026) and layer-dependent dynamics for adaptive cache refresh (Nguyen-Tri et al., 2026). Complementary approaches reduce denoising passes through confidence-aware parallel decoding, adaptive sampling stages, or continuous mixing of unresolved token states (Wu et al., 2026; Wei et al., 2026; Ye et al., 2026; Ben-Hamu et al., 2026; Israel et al., 2026; Chen et al., 2026; Wang et al., 2026). For diffusion multimodal models, visual-token pruning and merging reduce the cost of processing visual conditions (Xu et al., 2026; Chang et al., 2026). Within unified models, FlashU (Ke et al., 2026) introduces task-specific pruning and dynamic layer skipping on a hybrid architecture, while SPRINT (Bie et al., 2026) combines modality-aware prefix pruning with confidence-adaptive unmasking. Our work focuses on task-dependent computation within a shared masked diffusion backbone, coordinating channels, layers, and tokens throughout denoising.

3 Method

We first analyze task structural sparsity and token functional specialization within the shared network (Section 3.1) and introduce the masked diffusion formulation (Section 3.2). Building on this analysis, we develop acceleration strategies at the network-structure and token-computation levels: the former adjusts computation across FFN channels and network layers (Section 3.3), while the latter distinguishes feature refresh from generation progress and organizes their dynamic scheduling (Section 3.4). Finally, a shared execution plan coordinates these strategies, using generation progress and prediction states to restore necessary computation and rebuild the relevant caches (Section 3.5).

3.1 Motivation

3.1.1 Task Structural Sparsity

We first observe that shared FFN channels exhibit different importance distributions across tasks: despite using the same model parameters, image generation and language output tasks show marked differences in their channel importance rankings, and this divergence becomes more pronounced with increasing network depth. This indicates that different tasks develop different computational

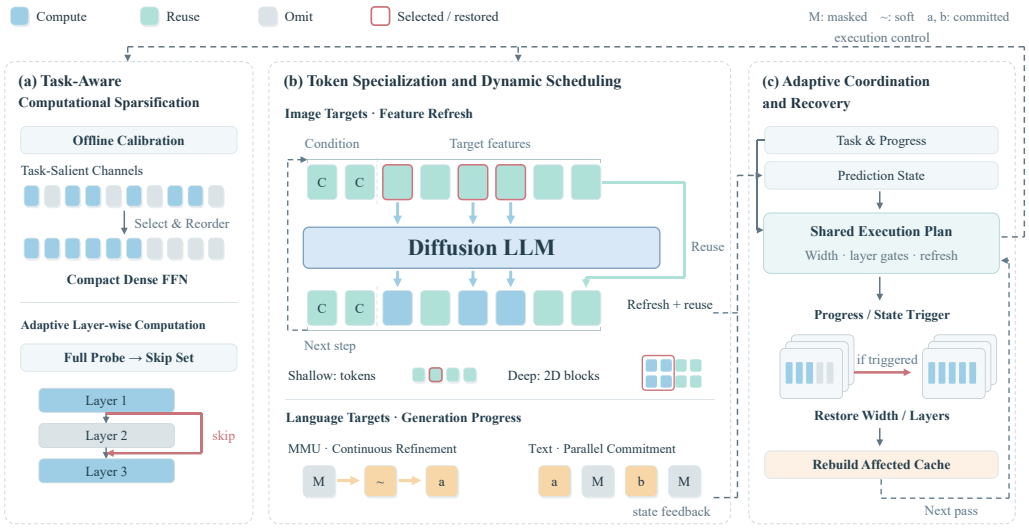


Figure 2: Overview of S^{2}I-dLLM. (a) Task-aware computational sparsification. (b) Token specialization and dynamic scheduling. (c) Adaptive coordination and recovery through generation-state feedback.

dependencies on shared channels, and that shared weights do not imply identical computational demands. Examining the depth dimension further, we find that these task differences also manifest in dependencies on computation across layers. From a computational perspective, shared parameters specify the same transformation rules, but the actual contributions of individual channels and layers also depend on the activation distributions induced by the task conditions and the current denoising state. Therefore, the importance of a computational unit is not a fixed property of its parameters alone, but a conditional property jointly determined by the task and the state. This offers an explanatory perspective on task structural sparsity: for a given task and denoising stage, some channels or layers may contribute only marginally to the current prediction, providing a basis for selectively retaining critical computation and restoring omitted computation as needed.

3.1.2 Token Functional Specialization

We further observe that tokens serving different functions exhibit different patterns of representation evolution during denoising. Although the content of conditioning tokens is already given, bidirectional attention still allows them to interact with the content being generated, so their hidden representations are not entirely unchanged; compared with target tokens under generation, their changes across steps are typically smaller, exhibiting relatively stable representations (Figure 3(a)). This indicates that providing contextual conditions and forming generated outputs correspond to different computational demands, and that tokens should not be treated as requiring equal computation merely because they pass through the same network.

Focusing on target tokens under generation, we find that the organization of their representation changes also varies across tasks. Image target tokens are arranged on a regular two-dimensional grid, with their changes across steps exhibiting spatial clustering and more pronounced local correlations in deeper layers (Figure 3(b)); language target tokens, in contrast, are organized as a one-dimensional sequence, with changes scattered across a small number of undetermined positions, making it difficult for comparable regular spatial blocks to emerge. This indicates that even among generation targets, tokens in different tasks exhibit different structures of representation evolution, leading to different requirements for computational granularity and refresh timing.

Together, these observations reveal token functional specialization: the computational demands of tokens depend not only on their modality, but also closely on whether they serve as conditions or generation targets, as well as on the current denoising state. Even within the image modality, tokens

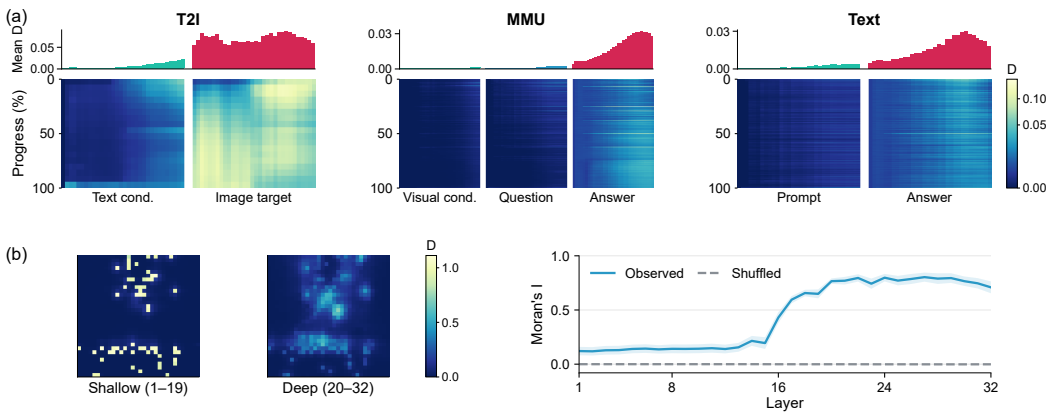


Figure 3: Token functional specialization during denoising. (a) Cross-step feature drift by token role: conditioning representations remain relatively stable, while target representations continue to evolve. (b) Spatial distributions of image-target drift and shuffled-position controls, showing stronger local spatial correlations in deeper layers.

are continuously evolving targets in image generation, whereas they participate in computation as given visual conditions in multimodal understanding. Therefore, modality identity alone is insufficient to determine computation allocation. The relative stability of representations provides a basis for identifying candidates for feature reuse, while the local correlations and ongoing evolution of target tokens provide a basis for selective refresh and task-dependent organization of computation.

3.2 Preliminaries

Given the condition c_τ and target sequence x_0 for task τ, we independently mask each target token with probability u to obtain x_u. The corresponding time-weighted masked cross-entropy objective is (Sahoo et al., 2024; Wu et al., 2026):

$$\mathcal{L}_\tau(\theta) = \mathbb{E}_{(x_0,c_\tau)\sim\mathcal{D}_\tau} \int_0^1 \frac{1}{u} \mathbb{E}_{x_u\sim q_{u|0}(\cdot|x_0)} \left[\sum_{i\in\mathcal{M}_u} -\log p_\theta(x_0^i \mid x_u, c_\tau) \right] \mathrm{d}u, \qquad \mathcal{M}_u = \{i : x_u^i = [\mathrm{MASK}]\}. \tag{1}$$

Here $\mathcal{D}_\tau$ is the task data distribution, $\mathcal{M}_u$ denotes the masked positions, and $u \in (0, 1]$ is the masking level. S^{2}I-dLLM keeps the model parameters fixed and adjusts inference computation according to the task and denoising state.

3.3 Task-Aware Computational Sparsification

Different tasks exhibit different dependencies on shared channels and network layers. Inspired by Wanda and FlashU (Sun et al., 2024; Ke et al., 2026), we use a single small-scale offline calibration to obtain the activations $a_{l,j}^{(\tau)}$ of the j-th intermediate FFN channel in layer l over the target tokens of task τ, and combine them with the corresponding output projection weights $w_{l,j}^{\mathrm{out}}$ to define the channel saliency score as:

$$s_{l,j}^{(\tau)} = \left\| a_{l,j}^{(\tau)} \right\|_2 \left\| w_{l,j}^{\mathrm{out}} \right\|_1, \tag{2}$$

Based on the saliency scores $s_{l,j}^{(\tau)}$ in Equation equation 2, we select the $k_{\tau,l}$ highest-scoring channels in each layer to form the index set $\mathcal{K}_{\tau,l}$, and jointly slice the three SwiGLU projections accordingly:

$$\mathrm{FFN}_{l,\mathrm{c}}^{(\tau)}(x) = W_l^o[:, \mathcal{K}_{\tau,l}] \left[\mathrm{SiLU}(W_l^g[\mathcal{K}_{\tau,l}, :]x) \odot W_l^u[\mathcal{K}_{\tau,l}, :]x \right]. \tag{3}$$

The retained channels indexed by $\mathcal{K}_{\tau,l}$ are consistently reordered into a contiguous prefix, directly translating task-dependent channel selection into narrower dense computation.

We further use full probes to record the hidden representations $h^{\text{in}}_{l,t,q}$ and $h^{\text{out}}_{l,t,q}$ before and after the l-th Transformer block at target position $q \in \Omega_\tau$ during denoising step t, and compute their mean cosine similarity over the target region to identify candidate redundant layers:

$$S^{(\tau)}_{l,t} = \frac{1}{|\Omega_\tau|} \sum_{q \in \Omega_\tau} \cos\big(h^{\text{in}}_{l,t,q}, h^{\text{out}}_{l,t,q}\big)\,, \tag{4}$$

Here Ω_τ denotes the set of positions of image or answer tokens to be generated, excluding the conditioning positions $\mathcal{C}_\tau$ and padding positions $\mathcal{P}_\tau$. A higher similarity indicates a smaller directional transformation. We select a preset fraction r_{LS} of layers with the highest similarity to form the candidate skip set $\mathcal{L}^{(\tau,t)}_{\text{skip}}$, and apply bypassing through a computation gate:

$$h_{l+1,t} = h_{l,t} + m_{l,t}\big(\mathcal{B}_l(h_{l,t}) - h_{l,t}\big). \tag{5}$$

Here $\mathcal{B}_l$ denotes the l-th Transformer block; when $l \in \mathcal{L}^{(\tau,t)}_{\text{skip}}$, $m_{l,t} = 0$ and the layer executes the identity mapping; otherwise, $m_{l,t} = 1$ and the full block is evaluated. A full probe is performed at the first step, after which the skip set is updated every T_{LS} steps, with the most recent selection reused in between. Phase-aware width recovery and on-demand layer recomputation further mitigate approximation errors from sustained computational reduction, as detailed in Section 3.5.

3.4 Token Specialization and Dynamic Scheduling

Token functional specialization implies that different positions have different requirements for repeated computation and continued refinement. We therefore distinguish feature refresh from generation progress: the former uses representational changes to determine which intermediate computations need updating, while the latter uses prediction states to determine which positions require further refinement or commitment.

For feature refresh, we organize cross-step changes into computational units that match the representation structure. Image targets use individual tokens as units in shallow layers and local two-dimensional blocks in deeper layers where spatial correlations are more pronounced; the refresh scope is then adaptively determined from the aggregated change within each unit. Selected positions are recomputed, while the remaining positions reuse cached features, with conditioning representations updated through periodic full computation (Liu et al., 2025). This allows the computation granularity and refresh scope to adapt to network depth and the current generation state, respectively.

For generation progress, we distinguish the continuous propagation of predictive information from discrete commitment. Undetermined positions retain partial semantic information through soft predictions to participate in subsequent refinement (Ye et al., 2026); positions that meet the confidence requirement are committed in parallel, reducing unnecessary generation rounds (Wu et al., 2026). In the current implementation, multimodal understanding uses continuous state refinement and text reasoning uses confidence-gated commitment, both integrated with generation-state feedback and computation recovery. Further details are provided in Appendix B.2.

3.5 Adaptive Coordination and Recovery

We coordinate FFN width, the set of executed layers, and cache refresh through a shared execution plan, restoring necessary computation as the generation state evolves.

Following the notation in Section 3.2, at the same input and denoising state, let $p_i = p_\theta(\cdot \mid x_u, c_\tau)_i$ denote the predictive distribution under full computation and $\widetilde{p}_i$ the predictive distribution obtained using the acceleration strategies above. For a masked position $i \in \mathcal{M}_u$, the cross-entropy change for its ground-truth target x^i_0 is:

Table 1: Multimodal understanding and image generation on MMaDA-8B-MixCoT. MMU uses token-only S^{2}I-dLLM and local judging. T2I latency uses 16 fixed samples per method; quality uses the full evaluation sets (Appendix A).

Dataset	Method	MMaDA-8B-MixCoT			
		NFE ↓	Latency (s) ↓	Score ↑	Speedup ↑
(a) Multimodal understanding					
Flickr30k-lite	Vanilla	256	46.23	57.49	1.00×
	+ dLLM-Cache	256	15.61	53.67	2.96×
	+ Fast-dLLM	30.84	1.86	52.96	24.79×
	+ ReMix	30.90	5.64	53.98	8.20×
Captioning	**S^{2}I-dLLM (ours)**	22.25	**1.54**	**54.32**	**30.04×**
AI2D-lite	Vanilla	256	47.32	67.20	1.00×
	+ dLLM-Cache	256	15.77	67.00	3.00×
	+ Fast-dLLM	52.02	3.15	66.80	15.00×
	+ ReMix	36.75	6.84	**67.60**	6.92×
Diagram understanding	**S^{2}I-dLLM (ours)**	39.31	**2.32**	66.20	**20.44×**
MathVista-mini	Vanilla	256	47.19	26.60	1.00×
	+ dLLM-Cache	256	15.69	26.40	3.01×
	+ Fast-dLLM	67.94	4.00	26.40	11.79×
	+ ReMix	48.41	8.95	27.30	5.27×
Math reasoning	**S^{2}I-dLLM (ours)**	52.04	**2.95**	**28.00**	**16.02×**
MMMU-val	Vanilla	256	49.02	29.33	1.00×
	+ dLLM-Cache	256	15.97	**28.67**	3.07×
	+ Fast-dLLM	74.95	4.43	27.89	11.07×
	+ ReMix	53.74	10.82	26.89	4.53×
Multi-discipline reasoning	**S^{2}I-dLLM (ours)**	59.62	**3.24**	27.44	**15.12×**
(b) Text-to-image generation					
GenEval	Vanilla	50	20.29	0.5404	1.00×
	Few-step Vanilla	32	12.96	**0.5518**	1.57×
	+ dLLM-Cache	50	9.13	0.4298	2.22×
	+ ML-Cache	50	11.59	0.5460	1.75×
Compositional alignment	**S^{2}I-dLLM (ours)**	35	**8.49**	0.5489	**2.39×**
DPG-Bench	Vanilla	50	21.59	70.10	1.00×
	Few-step Vanilla	32	13.77	70.04	1.57×
	+ dLLM-Cache	50	9.64	62.08	2.24×
	+ ML-Cache	50	12.15	69.46	1.78×
Dense-prompt alignment	**S^{2}I-dLLM (ours)**	35	**8.90**	**71.30**	**2.43×**

$$\Delta\ell_i = -\log \widetilde{p}_i(x_0^i) + \log p_i(x_0^i) = \log \frac{p_i(x_0^i)}{\widetilde{p}_i(x_0^i)}. \tag{6}$$

This quantity is used for offline analysis of computational sensitivity. Let a_i and $\widetilde{a}_i$ denote the logits output at position i by full computation and the acceleration strategies above, respectively, and define their maximum componentwise deviation as $\varepsilon_i = \|\widetilde{a}_i - a_i\|_\infty$. Since the gradient of log-sum-exp is softmax (Blanchard et al., 2021) and has unit ℓ_1 norm, log-sum-exp is 1-Lipschitz under the infinity norm, giving:

$$|\Delta\ell_i| \le 2\|\widetilde{a}_i - a_i\|_\infty = 2\varepsilon_i. \tag{7}$$

To mitigate prediction deviations caused by sustained computational reduction, we introduce phase-aware width recovery for the compact FFN in Equation equation 3. Let T denote the total number of denoising steps, $p_t = (t+1)/T$ the denoising progress, d_{ff} the full FFN width, and β the fraction of the trajectory assigned to full computation at the end. Using the step function H, we adjust the retained width $k_{\tau,l}$ as follows:

Table 2: Text reasoning on MMaDA-8B-MixCoT. S^{2}I-dLLM uses token scheduling only.

Dataset	Method	MMaDA-8B-MixCoT			
		NFE ↓	Latency (s) ↓	Score ↑	Speedup ↑
GSM8K	Vanilla	256	15.05	54.66	1.00×
0-shot	+ dLLM-Cache	256	10.09	**55.42**	1.49×
	+ Fast-dLLM	95.86	3.94	52.39	3.82×
	+ ReMix	74.38	4.30	53.68	3.50×
	S^{2}I-dLLM	82.53	**3.05**	54.06	**4.94×**
MATH-500	Vanilla	256	15.01	14.20	1.00×
0-shot	+ dLLM-Cache	256	10.07	13.20	1.49×
	+ Fast-dLLM	119.61	4.89	12.80	3.07×
	+ ReMix	94.00	5.41	11.00	2.77×
	S^{2}I-dLLM	107.21	**3.83**	**13.60**	**3.92×**
ARC-E	Vanilla	256	15.83	54.34	1.00×
0-shot	+ dLLM-Cache	256	10.12	**57.45**	1.56×
	+ Fast-dLLM	92.12	3.78	55.93	4.19×
	+ ReMix	87.95	5.30	54.21	2.99×
	S^{2}I-dLLM	90.40	**3.34**	56.57	**4.74×**
ARC-C	Vanilla	256	15.88	41.47	1.00×
0-shot	+ dLLM-Cache	256	10.19	43.48	1.56×
	+ Fast-dLLM	85.40	3.56	43.14	4.45×
	+ ReMix	80.73	4.90	42.14	3.24×
	S^{2}I-dLLM	86.05	**3.16**	**46.82**	**5.02×**

$$d_{\tau,l}(p_t) = k_{\tau,l} + \big(d_{\text{ff}} - k_{\tau,l}\big) H(p_t - (1-\beta)), \tag{8}$$

Computation uses the first $d_{\tau,l}(p_t)$ channels after consistent reordering: the compact FFN in Equation equation 3 is executed early on, and the full FFN is restored in the final phase.

For language generation with an unknown number of rounds, progress is measured by the fraction of committed positions, and full width and depth are restored in the final phase. Layer selection is updated through periodic full probes; when observed or predicted changes in target predictions significantly exceed the task prior, the corresponding layers are restored early. Initialization, full probes, and global recovery execute at full width and depth, and the relevant caches are rebuilt after layer restoration or a width switch.

This bound characterizes the local effect of computational approximation on prediction loss at a fixed denoising state, providing a basis for analyzing computation recovery. The detailed derivation and scope are given in Appendix B.3.

4 Experiments

4.1 Experimental Setup

We evaluate S^{2}I-dLLM on MMaDA-8B-MixCoT across multimodal understanding, image generation, and text generation, reporting task quality, forward evaluations (NFE), and mean per-sample latency. The main MMU and text comparisons use a fixed 256-position output canvas. Speedup is relative to the matched vanilla model; settings and metric definitions are given in Appendix A. In the main comparison tables, bold numerical entries mark the best values among accelerated methods; n/a denotes quantities not evaluated or not applicable.

4.2 Main Results

Token-only S^{2}I-dLLM achieves 15.12–30.04× speedups for multimodal understanding, with local-judge accuracy changes ranging from -1.89 to $+1.40$ percentage points and a 3.17 CIDEr drop on Flickr30k-lite (Table 1a). Across the main and additional results (Tables 1 and 6a), S^{2}I-dLLM

reduces latency relative to Vanilla on all six MMU benchmarks, while ReMix achieves higher quality on AI2D-lite and ScienceQA-IMG, highlighting task-dependent speed–quality trade-offs. With 35 forward evaluations, S^{2}I-dLLM achieves 2.39–$2.43\times$ acceleration on GenEval and DPG-Bench (Table 1b), and obtains the highest DPG and ImageReward scores among the five compared methods (Tables 1b and 6b). Few-step Vanilla retains a higher GenEval score at longer latency. The T2I comparisons measure the combined effect of fewer denoising calls and reduced computation within each call. Few-step Vanilla uses 32 calls and serves as a reduced-step reference; it does not isolate the computation mechanisms at matched NFE.

Token-only text acceleration reaches 3.92–$10.48\times$, with quality gains and losses across the reported benchmarks (Tables 2 and 6c). With model-specific calibration, cross-model evaluation yields 2.07–$4.71\times$ speedups with model-dependent quality trade-offs (Table 3).

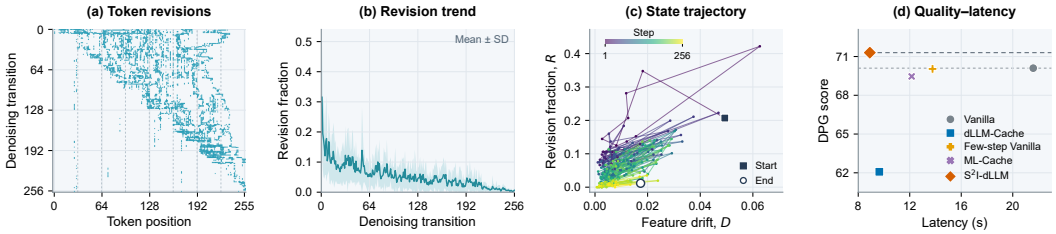


Figure 4: Token dynamics and quality–latency trade-offs. (a–c) Unaccelerated MMaDA-8B-Base: full-canvas prediction revisions, revision rate (mean ± SD over eight inputs), and feature-drift/revision trajectories. Panels (a,c) use the first input; revisions are not commits. (d) DPG-Bench quality and latency reported in Table 1b for MMaDA-8B-MixCoT; the dashed frontier marks the best reported quality within each latency budget.

Table 3: Cross-model evaluation with model-specific calibration on fixed subsets.

Benchmark	Base ↑	S^{2}I-dLLM ↑	Δ	Speedup ↑
Dynin-Omni				
AI2D-Lite	47.92	48.96	+1.04	4.71×
GenEval	0.8490	0.8385	−0.0104	2.07×
GSM8K	80.47	79.69	−0.78	3.60×
Lumina-DiMOO				
AI2D-Lite	43.75	42.71	−1.04	2.54×
GenEval	0.8281	0.8229	−0.0052	2.19×

Table 4: DPG-Bench ablations (35 NFE).

Strategy	Time (s) ↓	Speedup ↑	Score ↑
S^{2}I-dLLM (full)	8.90	2.43×	71.30
w/o Computational Sparsification	10.62	2.03×	71.58
w/o Feature Reuse	12.87	1.68×	71.46
Fixed Recovery	9.24	2.34×	70.96

4.3 Ablation Studies

Appendix C.3 compares channel ranking, layer selection, refresh granularity, and answer-state backends under the stated controls. Table 4 outlines a 35-NFE DPG-Bench ablation design on MMaDA-8B-MixCoT. Fixed Recovery retains recovery but replaces state-triggered layer restoration with a preset schedule.

5 Conclusion

We identified task structural sparsity and token functional specialization in multimodal dLLMs. Building on these findings, S^{2}I-dLLM provides training-free, task-aware acceleration by coordinating computational sparsification, dynamic token scheduling, and recovery across denoising states. Experiments on multimodal understanding, image generation, and text reasoning show reduced inference costs with task-dependent efficiency–quality trade-offs. Cross-model evaluations with model-specific recalibration further demonstrate applicability across backbones.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript drafting, language editing, document formatting and organization, and literature search and synthesis. These tools also assisted with code organization, refactoring, and integration. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study uses existing models and public benchmarks, with a focus on inference efficiency. Model outputs may still contain errors or biases, and their reliability should be assessed for the specific application.

REPRODUCIBILITY STATEMENT

The supplementary materials provide the core implementation of S²I-dLLM for inspection, including its algorithmic modules and MMaDA integration hooks. The main paper and appendix describe the framework, experimental settings, evaluation protocols, and implementation details. End-to-end reproduction additionally requires the full evaluation pipeline, model weights, datasets, and calibration artifacts, which are not included in this code supplement.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, volume 38, pp. 55981–56007, 2026. doi: 10.52202/085713-1874. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/510e22f4a2da5212a64f1591736e2eaf-Abstract-Conference.html.

Tiwei Bie, Haoxing Chen, Tieyuan Chen, Zhenglin Cheng, Long Cui, Kai Gan, Zhicheng Huang, Zhenzhong Lan, Haoquan Li, Jianguo Li, Tao Lin, Qi Qin, Hongjun Wang, Xiaomei Wang, Haoyuan Wu, Yi Xin, and Junbo Zhao. Llada2.0-uni: Unifying multimodal understanding and generation with diffusion large language model. *arXiv preprint arXiv:2604.20796*, 2026.

Pierre Blanchard, Desmond J. Higham, and Nicholas J. Higham. Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 41(4):2311–2330, 2021.

Shuochen Chang, Xiaofeng Zhang, Qingyang Liu, and Li Niu. D^{3}tom: Decider-guided dynamic token merging for accelerating diffusion mllms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 19961–19969, 2026. doi: 10.1609/aaai.v40i24.39080.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *International Conference on Learning Representations*, pp. 53193–53211, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/57250222014c35949476f3f272c322d2-Abstract-Conference.html.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, pp. 5046–5073, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0fa81c3f0d57f95b8776de3a248ef0ed-Abstract-Conference.html.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12454–12465,

2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/67d96d458abdef21792e6d8e590244e7-Abstract.html.

Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed S. Abdelfattah, Jae-sun Seo, Zhiru Zhang, and Udit Gupta. FlashDLM: Accelerating diffusion language model inference via efficient KV caching and guided diffusion. In *International Conference on Learning Representations*, pp. 153529–153544, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f930c6e1292a1160189a0734f22b465b-Abstract-Conference.html.

Jianuo Huang, Yaojie Zhang, Yicun Yang, Benhao Huang, and Linfeng Zhang. Mask tokens as prophet: Fine-grained cache eviction for efficient dLLM inference. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3456–3479, 2026. doi: 10.18653/v1/2026.findings-acl.170. URL https://aclanthology.org/2026.findings-acl.170/.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion LLMs via adaptive parallel decoding. In *Advances in Neural Information Processing Systems*, volume 38, pp. 52870–52888, 2026. doi: 10.52202/085713-1763. URL https://papers.neurips.cc/paper_files/paper/2025/hash/4c7d31b2da05a17f15101db5b37a1e24-Abstract-Conference.html.

Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, and Xu Yang. d^2cache: Accelerating diffusion-based LLMs via dual adaptive caching. In *International Conference on Learning Representations*, pp. 137511–137531, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/de4f2e3b8263aff39f0dd7571a8dc4b1-Abstract-Conference.html.

Junlong Ke, Zichen Wen, Boxue Yang, Yantai Yang, Xuyang Liu, Chenfei Liao, Zhaorun Chen, Shaobo Wang, and Linfeng Zhang. Flash-unified: A training-free and task-aware acceleration framework for native unified models. *arXiv preprint arXiv:2603.15271*, 2026.

Jaeik Kim, Woojin Kim, Jihwan Hong, Yejoon Lee, Sieun Hyeon, Mintaek Lim, Yunseok Han, Dogeun Kim, Hoeun Lee, Hyunggeun Kim, and Jaeyoung Do. Dynin-Omni: Omnimodal unified large diffusion language model. *arXiv preprint arXiv:2604.00007*, 2026. URL https://arxiv.org/abs/2604.00007.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, 1989.

Shufan Li, Jiuxiang Gu, Kangning Liu, Zhe Lin, Zijun Wei, Aditya Grover, and Jason Kuen. Lavida-o: Elastic large masked diffusion models for unified multimodal understanding and generation. In *International Conference on Learning Representations*, pp. 50571–50603, 2026a.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. Lavida: A large diffusion language model for multimodal understanding. In *Advances in Neural Information Processing Systems*, volume 38, pp. 105101–105134, 2026b. doi: 10.52202/085713-3511.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 149009–149033, 2026.

Quan Nguyen-Tri, Mukul Ranjan, and Zhiqiang Shen. Attention is all you need for KV cache in diffusion LLMs. In *International Conference on Learning Representations*, pp. 34915–34946, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/3afaa2102fb8ea44cbadc13e45bba718-Abstract-Conference.html.

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 50608–50646, 2026.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184, 2024. doi: 10.52202/079017-4135.

Qingyu Shi, Jinbin Bai, Zhuoran Zhao, Wenhao Chai, Kaidong Yu, Jianzong Wu, Shuangyong Song, Yunhai Tong, Xiangtai Li, Xuelong Li, and Shuicheng Yan. Muddit: Liberating generation beyond text-to-image with a unified discrete diffusion model. *arXiv preprint arXiv:2505.23606*, 2025.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Sparse-dLLM: Accelerating diffusion LLMs with dynamic cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33038–33046, 2026. doi: 10.1609/aaai.v40i39.40586. URL https://ojs.aaai.org/index.php/AAAI/article/view/40586.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Alexander Swerdlow, Mihir Prabhudesai, Siddharth Gandhi, Deepak Pathak, and Katerina Fragkiadaki. Unified multimodal discrete diffusion. *arXiv preprint arXiv:2503.20853*, 2025.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. Diffusion LLMs can do faster-than-AR inference via discrete diffusion forcing. In *International Conference on Learning Representations*, pp. 2282–2299, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/04bb76a98d9f32226b3beba7bd26a51f-Abstract-Conference.html.

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Puyu Zeng, Yuxuan Wang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast sampling: The three golden principles. In *International Conference on Learning Representations*, pp. 4340–4354, 2026.

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. In *International Conference on Learning Representations*, pp. 57027–57051, 2026.

Yi Xin, Qi Qin, Siqi Luo, Kaiwen Zhu, Juncheng Yan, Yan Tai, Jiayi Lei, Yuewen Cao, Keqi Wang, Yibin Wang, Jinbin Bai, Qian Yu, Dengyang Jiang, Yuandong Pu, Haoxing Chen, Le Zhuo, Junjun He, Gen Luo, Tianbin Li, Ming Hu, Jin Ye, Shenglong Ye, Bo Zhang, Chang Xu, Wenhai Wang, Hongsheng Li, Guangtao Zhai, Tianfan Xue, Bin Fu, Xiaohong Liu, Yu Qiao, and Yihao Liu. Lumina-dimoo: An omni diffusion large language model for multi-modal generation and understanding. *arXiv preprint arXiv:2510.06308*, 2025.

Jingqi Xu, Jingxi Lu, Chenghao Li, Sreetama Sarkar, Souvik Kundu, and Peter A. Beerel. Redvtp: Training-free acceleration of diffusion vision-language models inference via masked token-guided visual token pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2783–2792, 2026.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 138867–138907, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Yushi Ye, Feng Hong, Huangjie Zheng, Xu Chen, Zhiyong Chen, Yanfeng Wang, and Jiangchao Yao. Rejection mixing: Fast semantic propagation of mask tokens for efficient dllm inference. *arXiv preprint arXiv:2602.22868*, 2026.

Zebin You, Shen Nie, Xiaolu Zhang, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10093–10105, 2026.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion multimodal large language model with parallel decoding. *arXiv preprint arXiv:2505.16990*, 2025.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. LLaDA-MoE: A sparse MoE diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025. URL https://arxiv.org/abs/2509.24389.

A EXPERIMENTAL SETUP

We evaluate S^{2}I-dLLM on multimodal understanding (MMU), text reasoning, and text-to-image generation (T2I), measuring task quality, the number of forward evaluations (NFE), and mean request latency. The main comparisons use a shared backbone, with additional experiments assessing transfer to other multimodal dLLMs.

A.1 MODELS AND BASELINES

Backbone models. All main-table methods use MMaDA-8B-MixCoT (Yang et al., 2026), keeping the model checkpoint fixed across acceleration methods. Cross-model experiments use Dynin-Omni (Kim et al., 2026) and Lumina-DiMOO (Xin et al., 2025); each accelerated model is compared with its own vanilla inference configuration.

MMU and text-reasoning baselines. We compare with vanilla masked diffusion decoding, dLLM-Cache (Liu et al., 2025), Fast-dLLM (Wu et al., 2026), and ReMix (Ye et al., 2026). ReMix uses a confidence threshold of 0.80, a mixing coefficient of 0.40, and a Jensen–Shannon divergence threshold of 0.10. Fast-dLLM uses its v1 dual-cache implementation with confidence threshold 0.90. For dLLM-Cache, the prompt/target refresh intervals are 20/5 for MMU and 20/2 for text reasoning, with transfer ratio 0.25.

Image-generation baselines. We compare with Vanilla, Few-step Vanilla, dLLM-Cache, and ML-Cache adapted from Lumina-DiMOO (Xin et al., 2025). Vanilla, dLLM-Cache, and ML-Cache use 50 forward evaluations, whereas Few-step Vanilla uses 32 without caching. The dLLM-Cache refresh intervals are 5/5 with transfer ratio 0.25. ML-Cache uses cache ratio 0.90, warm-up ratio 0.30, and refresh interval 5. Its adaptation retains MMaDA's sampler, image vocabulary, padding mask, and separate conditional and unconditional cache states.

A.2 DATASETS AND EVALUATION METRICS

Multimodal understanding. We evaluate six benchmarks in the zero-shot setting:

- **Flickr30k-lite** contains 500 image-captioning examples and evaluates visual description.
- **AI2D-lite** contains 500 questions on scientific diagrams and evaluates diagram understanding.
- **MathVision-testmini** and **MathVista-testmini** contain 304 and 1,000 examples, respectively, and evaluate mathematical reasoning grounded in visual inputs.
- **MMMU-val** contains 900 examples spanning multiple disciplines.
- **ScienceQA-IMG** contains 2,017 science questions with image inputs.

Flickr30k-lite is scored with PTB-tokenized CIDEr on complete raw responses, reported as CIDEr $\times 100$. The other five benchmarks report accuracy under the same local scoring protocol for all methods: explicit, unambiguous final answers are matched directly, and remaining responses are evaluated using Qwen3-14B with BF16 computation, non-thinking mode, and greedy decoding. MathVista retains the original normalization and matching rules after local answer extraction; extractions unsupported by the raw response are treated as unanswered.

Text reasoning. The text suite uses matched prompts and output lengths across methods and covers four groups:

- **Mathematical reasoning:** GSM8K (1,319 examples) and MATH-500 (500).
- **Code generation:** MBPP (500), evaluated with pass@1.
- **Logical reasoning:** Countdown (256).
- **Science question answering:** ARC-E test (2,376) and ARC-C validation (299).

All reported text benchmarks use zero-shot evaluation. Mathematical reasoning, Countdown, and ARC report answer accuracy.

Text-to-image generation. We evaluate compositional alignment, dense-prompt following, and preference scores:

- **GenEval** uses 553 prompts to assess object presence, counting, colors, spatial relations, and attribute binding, with scores on a 0–1 scale.
- **DPG-Bench** uses 1,065 dense prompts to evaluate fine-grained prompt adherence, with scores on a 0–100 scale.
- **ImageReward-v1.0** evaluates the same DPG images using the official 35-token text limit.

Each method generates four images per prompt, yielding 2,212 GenEval and 4,260 DPG images. All methods share the same local evaluators; DPG and ImageReward scoring use FP32.

Cross-model evaluation. To assess the applicability of S^{2}I-dLLM across multimodal dLLMs, we select Dynin-Omni (Kim et al., 2026) and Lumina-DiMOO (Xin et al., 2025), two open-source models with unified multimodal understanding and generation capabilities, and compare each accelerated model with its original inference configuration. Both models are evaluated on 96 AI2D-Lite questions and 48 GenEval prompts with four seeds per prompt; Dynin-Omni is additionally evaluated on 128 GSM8K questions. For each task, eight calibration and eight development examples are disjoint from the evaluation subset. GenEval contains eight prompts per category and reports the six-category macro average. AI2D uses strict multiple-choice answer matching, while GSM8K uses numerical-answer accuracy. These subsets use fixed settings selected before evaluation.

The extended evaluation also covers all 2,017 ScienceQA-IMG and 900 MMMU-val questions for both models. Calibration remains frozen. MMU retains a 16-token answer budget; the 39 multi-image MMMU questions use numbered image panels through each model's single-image input path. Answer text is delimited by the native answer tags before benchmark parsing, consistently for both methods; the same correction is applied to the archived Lumina AI2D outputs. MMMU retains the reference parser's per-sample seeded fallback, with no-fallback scores stored separately. These results measure matched transfer configurations rather than each model's best native benchmark setting.

Except for the metrics specified above, scores are reported as percentages. Score differences use the corresponding metric units, and speedups are computed from unrounded mean latencies unless stated otherwise in the table caption.

A.3 Experimental Configuration and Timing Protocol

Offline calibration. Offline calibration computes task-specific FFN channel rankings using the saliency scores in Equation (2), without updating model weights. The resulting channel permutation is reused during inference, while layer rankings are updated by full probes. Cross-model evaluation recalibrates the channel rankings separately for each backbone using the calibration subsets described above.

Hardware and decoding settings. The main experiments use NVIDIA RTX 4090 GPUs, batch size 1, and BF16 model computation. MMU and text reasoning use a fixed canvas of 256 output positions, with block lengths of 128 and 64, respectively. Text evaluation uses three GPUs capped at 450W, temperature 0, and no early EOS termination. T2I uses 512×512 resolution and classifier-free guidance scale 3.5, with seed $10086 + 4j + k$ for prompt index j and image index k. One T2I NFE denotes one backbone call with the conditional and unconditional branches batched together.

Latency measurement. We report mean per-request wall-clock latency. Speedup is the mean Vanilla latency divided by that of the compared method. Timing includes preprocessing, data transfers, generation, scheduling, and output decoding, while excluding model loading, offline calibration, warm-up, hook or adapter setup/teardown, file I/O, scoring, and cooling waits. MMU combines synchronized generation and transfer time with the fixed preprocessing cost recorded for each input. Thermally throttled attempts in the main comparisons are rerun with the same seed.

MMU and text comparison protocol. For text reasoning, Vanilla, dLLM-Cache, Fast-dLLM, and S^{2}I-dLLM run sequentially on the same physical GPU for each input, with method order rotated

across inputs to mitigate the effects of device differences and execution order on timing. ReMix is evaluated in a subsequent run with identical input shards, GPU assignments, and timing rules, using the original Vanilla mean as reference. For MMU, the three added baselines likewise rotate order on the same GPU per input; Vanilla and S^{2}I-dLLM retain earlier runs with the same checkpoint, inputs, and generation budget. All accepted timings for the added MMU baselines have zero thermal-throttling counter increments. In text evaluation, Vanilla, dLLM-Cache, and Fast-dLLM compute confidence in FP64, while S^{2}I-dLLM uses BF16.

Image-generation timing. Quality is evaluated on the complete image sets, whereas latency is measured separately on eight fixed prompts and two seeds per prompt for each benchmark, totaling 16 samples per method. All methods use the same physical GPU for a given benchmark. Timing includes VQ decoding and output transfer to CPU. ImageReward evaluates the generated DPG images without additional image generation.

Cross-model and ablation settings. Cross-model speedups use paired requests with matched inputs and seeds on the same GPU, following the resident-model timing scope above. AI2D uses 16 output tokens and block length 16; Dynin-Omni GSM8K uses 256 output tokens and block length 64. GenEval retains each backbone's generation settings in both arms.

B Implementation Details and Theoretical Supplement

B.1 Unified Inference Procedure

The task τ determines the channel ranking, target region Ω_τ, and token scheduling path. Offline calibration computes the saliency scores in Equation (2); the same channel permutation is applied to the gate, up, and output projections, retaining the full weights for later recovery. During inference, a shared execution plan specifies the FFN width, executed layers, and cache refreshes for each forward pass. We use f to index actual backbone calls; for subsampled image generation, the original denoising-grid index can differ from f. Algorithm 1 describes the joint framework; the configurations used in the main comparisons are listed in Section B.4.

Algorithm 1 S^{2}I-dLLM Inference

Require: Model p_θ, task τ, condition c_τ, calibrated rankings and settings
Initialize target state X, feature cache $\mathcal{Z} \leftarrow \emptyset$
Initialize prediction history $\mathcal{H} \leftarrow \emptyset$, forward index $f \leftarrow 0$
while generation is incomplete and the task budget remains **do**
  $p_f \leftarrow \textsc{Progress}(\tau, X, f)$
  $\Pi_f \leftarrow \textsc{Plan}(\tau, p_f, \mathcal{H})$ ▷ Width, layers, refresh
  if $f = 0$ or full probe due or final recovery active **then**
    $\Pi_f \leftarrow \textsc{FullCompute}(\Pi_f)$
  end if
  $\mathcal{Z} \leftarrow \textsc{InvalidateAffected}(\mathcal{Z}, \Pi_f)$
  $(P_f, \mathcal{Z}) \leftarrow \textsc{Forward}(p_\theta, X, c_\tau, \Pi_f, \mathcal{Z})$
  if full probe performed **then**
    Update $\mathcal{L}_{\text{skip}}$ using Equation (4)
  end if
  if $\tau = \text{T2I}$ **then**
    $X \leftarrow \textsc{ImageStep}(X, P_f)$
  else
    $X \leftarrow \textsc{RefineAndCommit}(\tau, X, P_f)$ ▷ Section B.2
  end if
  $\mathcal{H} \leftarrow \textsc{Observe}(\mathcal{H}, P_f)$ ▷ Feedback for the next pass
  $f \leftarrow f + 1$
end while
return $\textsc{Decode}(X)$

Here X includes target tokens and any soft embeddings, P_f is the current prediction, and Π_f is the execution plan. PLAN uses the latest layer ranking and prior prediction feedback; INVALIDATEAFFECTED marks caches for rebuilding after path changes or scheduled refreshes. FORWARD rebuilds

those entries and applies image refresh selection within each executed layer as described in Section B.2.

Recovery decisions use only information available before the affected forward pass. In particular, changes observed in the current prediction inform later decisions; they do not retroactively alter the pass that produced that prediction. Probe rankings are obtained with full width and all layers, and skipped blocks use the identity mapping in Equation (5).

B.2 Task-Specific Execution Paths

Token representation stability and predictive uncertainty imply different computational requirements. We therefore distinguish feature refresh from generation progress: the former identifies positions that require recomputation, while the latter determines how undetermined positions are refined and committed. Figure 5 illustrates feature refresh, continuous refinement, and parallel commitment for the three task paths.

B.2.1 Image Feature Refresh

Cross-step changes in image target tokens exhibit spatial correlations that become more pronounced in deeper layers. We therefore use individual tokens as refresh units in shallow layers and non-overlapping two-dimensional blocks in deeper layers. Using changes in value projections to assess cache validity (Liu et al., 2025), we define the positional change and unit score for layer l at forward pass f as:

$$\begin{aligned} d_i^{(l,f)} &= \max\Big(0,\, 1 - \cos\Big(V_i^{(l,f)}, \widetilde{V}_i^{(l,f)}\Big)\Big), \\ s_g^{(l,f)} &= \sum_{i \in g} d_i^{(l,f)}, \end{aligned} \tag{9}$$

where $\widetilde{V}$ denotes the cached value projection, $\mathcal{G}_l$ is the partition into refresh units, and $g \in \mathcal{G}_l$ denotes one unit. We sort units in descending order of their scores and select the smallest prefix that covers a fraction η of the total change:

$$\begin{aligned} m^* &= \min\left\{ m : \sum_{a=1}^{m} s_{g_{(a)}}^{(l,f)} \geq \eta \sum_{g \in \mathcal{G}_l} s_g^{(l,f)} \right\}, \\ \mathcal{R}_{l,f} &= \bigcup_{a=1}^{m^*} g_{(a)}. \end{aligned} \tag{10}$$

This selection adapts the refresh granularity to network depth and the refresh scope to the current distribution of changes. Value projections are computed at all image positions, while only positions in $\mathcal{R}_{l,f}$ update their keys, attention outputs, and FFN outputs; the remaining positions reuse cached features, and conditioning tokens are refreshed on the first, periodic, and final full forward passes. For classifier-free guidance, the two branches share refresh positions selected using their maximum per-position change, while their cached feature values remain separate. Full refreshes and cache rebuilding override partial selection.

B.2.2 Continuous Refinement for Multimodal Understanding

For undetermined language targets, computational requirements also depend on how predictive information propagates between positions. For multimodal understanding, we retain predictive information at uncommitted positions through continuous embeddings (Ye et al., 2026). Let $q_{i,f}$ denote the stored prediction at position i after pass f, and $\mathcal{S}_{i,f}$ its selected vocabulary subset. The retained probability mass is $r_{i,f} = \sum_{v \in \mathcal{S}_{i,f}} q_{i,f}(v)$; the residual mass is assigned to the mask embedding rather than renormalized away. The next input embedding is:

$$z_{i,f+1} = (1 - \alpha r_{i,f}) e_{\text{mask}} + \alpha \sum_{v \in \mathcal{S}_{i,f}} q_{i,f}(v) e_v, \tag{11}$$

where α is the mixing coefficient, and e_{mask} and e_v are the mask and vocabulary embeddings. Positions without a usable history retain the mask embedding. The vocabulary subset is formed from

high-probability candidates. Positions exceeding the confidence threshold are committed subject to a per-pass cap; if none qualifies, the most confident position is committed. For positions that remain undetermined, a Jensen–Shannon divergence test resets unstable prediction histories before their next mixing update. In the main MMU configuration, feature reuse applies to the given image/question prefix within a decoding block, with a full prefix rebuild at block boundaries; answer-position features are recomputed on each pass.

B.2.3 Confidence-Guided Text Reasoning

For text reasoning, a fixed number of commitments cannot readily adapt to differing levels of certainty across positions. Based on predictive confidence, we select positions for parallel commitment from the uncommitted set $\mathcal{U}_f$ in the current block (Wu et al., 2026):

$$\mathcal{C}_f = \left\{ i \in \mathcal{U}_f : \max_{v \in \mathcal{V}} p_{i,f}(v) \geq \gamma \right\}, \tag{12}$$

where $p_{i,f}$ is the predictive distribution and γ the confidence threshold. If the set is empty, the most confident position is committed; this process continues until all positions in the current block are determined. The main text configuration combines this commitment rule with prefix caching, while recomputing answer-position features. Generation proceeds block by block until the prescribed target length is filled.

These decisions respectively adjust the computation scope within a forward pass and the rate of generation progress, with computation recovery coordinated through the mechanism in Section 3.5.

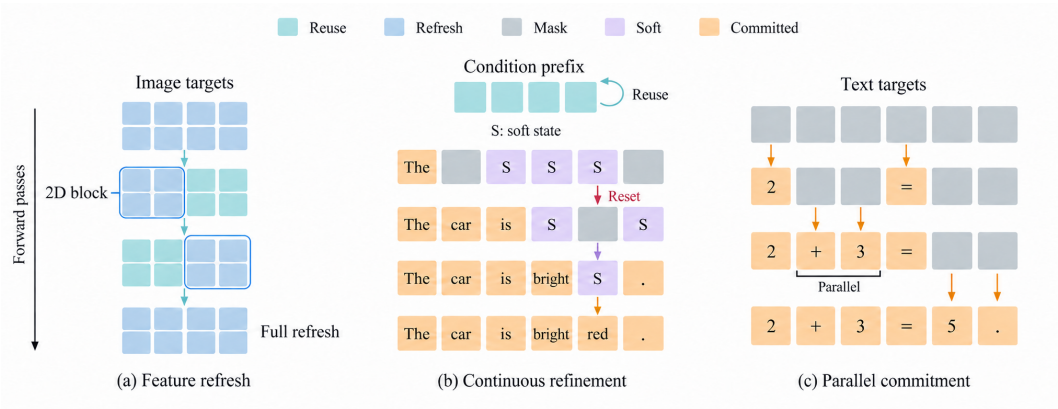


Figure 5: Token scheduling mechanisms: (a) feature refresh and reuse, (b) continuous refinement, and (c) parallel commitment.

B.3 Local Cross-Entropy Perturbation Bound

Following Section 3.2, fix the input x_u, condition c_τ, and masked position $i \in \mathcal{M}_u$. Let a_i and $\widetilde{a}_i$ be the logits under full and reduced computation, and define $\varepsilon_i = \|\widetilde{a}_i - a_i\|_\infty$. For the target x_0^i, write $\ell(a_i, x_0^i) = -a_{i,x_0^i} + \mathrm{LSE}(a_i)$, where $\mathrm{LSE}(a) = \log \sum_v e^{a_v}$. The componentwise bound implies:

$$e^{-\varepsilon_i} \sum_v e^{a_{i,v}} \leq \sum_v e^{\widetilde{a}_{i,v}} \leq e^{\varepsilon_i} \sum_v e^{a_{i,v}}. \tag{13}$$

Taking logarithms yields $|\,\mathrm{LSE}(\widetilde{a}_i) - \mathrm{LSE}(a_i)| \leq \varepsilon_i$, consistent with the standard log-sum-exp property (Blanchard et al., 2021). Thus:

$$\begin{aligned} |\Delta \ell_i| &= |\ell(\widetilde{a}_i, x_0^i) - \ell(a_i, x_0^i)| \\ &\leq |\widetilde{a}_{i,x_0^i} - a_{i,x_0^i}| + |\,\mathrm{LSE}(\widetilde{a}_i) - \mathrm{LSE}(a_i)| \\ &\leq 2\varepsilon_i. \end{aligned} \tag{14}$$

The same argument applies when reduced computation uses channel selection, layer bypassing, feature reuse, or their combination: their joint effect is represented by the resulting logit perturbation. This supplies a common local quantity for comparing computational approximations.

Scope. The bound concerns the same input state and does not establish a bound on cross-step trajectory errors or final task quality. Feature cosine similarity and predictive activity are scheduling proxies, not proven upper bounds on ε_i. Their relationship to prediction perturbations and the benefit of recovery must be evaluated empirically. Ground-truth targets are used only in offline loss analysis; online scheduling and recovery use generation-state feedback.

B.4 Hyperparameters and Default Settings

Table 5 summarizes the core hyperparameters for the main T2I comparisons and the joint MMU/text analyses. The joint configurations disable prefix reuse and use three initial full-computation passes. The main MMU and text comparisons instead use full FFN width and all layers with token scheduling and prefix caching; their commitment thresholds are 0.80 and 0.85, respectively.

Table 5: Core hyperparameters of S^{2}I-dLLM: (a) main T2I comparisons; (b) joint MMU and text analyses.

(a) Image generation

Hyperparameter	Value
FFN width reduction ratio	0.20
Candidate layer bypass ratio	0.20
Full-probe interval	10
Refresh interval	4
Change coverage η	0.90
Final recovery fraction β	0.10
Forward-call budget	35

(b) MMU and text reasoning

Hyperparameter	MMU	Text
FFN width reduction ratio	0.2	0.15
Candidate layer bypass ratio	0.10	0.10
Full-probe interval	10	10
Final recovery fraction β	0.30	0.30
Output length	256	256
Decoding block length	128	64
Commitment threshold γ	0.80	0.90
Mixing coefficient α	0.40	—
JS rejection threshold	0.10	—

For the main T2I results, S^{2}I-dLLM retains 35 points from the original 50-step denoising grid: indices 0–4, $5 + \mathrm{round}(38j/23)$ for $j = 0, \ldots, 23$, and 44–49. Sampling temperature and structural recovery follow the original grid, while cache refresh and probe intervals count executed forward passes. The initial five and final six passes use full computation. Pruning and skipping ratios are nominal budgets. One T2I NFE denotes one joint conditional/unconditional backbone call; the two guidance branches are batched together.

C Additional Experiments and Sensitivity Analysis

C.1 Additional Benchmark Results

Table 6 supplements the main comparisons with multimodal understanding (a), image generation (b), and text reasoning and code generation (c), using the same configurations and evaluation protocols. ImageReward scores the generated DPG-Bench images.

C.2 Cross-Model Evaluation

Table 3 evaluates applicability to two additional multimodal dLLMs with model-specific recalibration and no training.

While Table 3 provides a compact overview, Table 7 adds ScienceQA-IMG and MMMU-val results for Dynin-Omni and Lumina-DiMOO, covering all 2,017 and 900 questions, respectively. AI2D-lite retains the fixed subset above and repeats the scores and speedups from Table 3. Each model's Vanilla and S^{2}I-dLLM configurations are compared; Score denotes accuracy (%).

For ScienceQA-IMG and MMMU-val in Table 7, speedups range from 1.87 to $4.86\times$, with accuracy changes between -0.67 and $+0.15$ percentage points.

Table 6: Additional benchmark results on MMaDA-8B-MixCoT.

Dataset	Method	MMaDA-8B-MixCoT			
		NFE ↓	Latency (s) ↓	Score ↑	Speedup ↑
(a) Multimodal understanding					
MathVision-mini	Vanilla	256	47.47	17.43	1.00×
	+ dLLM-Cache	256	15.67	16.45	3.03×
	+ Fast-dLLM	76.16	4.44	15.79	10.70×
	+ ReMix	50.05	9.37	13.16	5.07×
Math reasoning	S^{2}I-dLLM (ours)	54.55	**2.98**	**16.78**	**15.92×**
ScienceQA-IMG	Vanilla	256	44.22	52.11	1.00×
	+ dLLM-Cache	256	15.72	53.69	2.81×
	+ Fast-dLLM	54.81	3.29	50.62	13.42×
	+ ReMix	41.71	7.79	**54.29**	5.67×
Multi-discipline reasoning	S^{2}I-dLLM (ours)	45.56	**2.43**	52.26	**18.17×**
(b) Text-to-image generation					
ImageReward (DPG)	Vanilla	50	21.59	0.2059	1.00×
	Few-step Vanilla	32	13.77	0.2093	1.57×
	+ dLLM-Cache	50	10.38	-0.3751	2.08×
	+ ML-Cache	50	12.15	0.1801	1.78×
Human preference	S^{2}I-dLLM (ours)	35	**9.64**	**0.2237**	**2.24×**
(c) Text reasoning and code generation					
MBPP	Vanilla	256	15.17	13.80	1.00×
0-shot	+ dLLM-Cache	256	10.13	13.20	1.50×
	+ Fast-dLLM	41.20	1.76	11.40	8.62×
	+ ReMix	39.73	2.32	**14.20**	6.54×
	S^{2}I-dLLM	35.07	**1.45**	13.20	**10.48×**
Countdown	Vanilla	256	16.10	16.41	1.00×
0-shot	+ dLLM-Cache	256	10.18	21.88	1.58×
	+ Fast-dLLM	79.98	3.36	19.14	4.79×
	+ ReMix	76.41	4.73	13.28	3.41×
	S^{2}I-dLLM	59.77	**2.35**	**24.61**	**6.86×**

Table 7: Extended cross-model evaluation on Dynin-Omni (left) and Lumina-DiMOO (right). Repeated scores and speedups follow Table 3; other speedups use the displayed latencies. Latency is in seconds.

Dynin-Omni

Benchmark	Method	Latency ↓	Speed ↑	Score ↑
AI2D-lite	Vanilla	2.19	1.00×	47.92
	S^{2}I-dLLM	0.46	4.71×	48.96
ScienceQA-IMG	Vanilla	2.09	1.00×	58.21
	S^{2}I-dLLM	0.43	4.86×	57.81
MMMU-val	Vanilla	2.19	1.00×	58.22
	S^{2}I-dLLM	0.49	4.47×	58.33

Lumina-DiMOO

Benchmark	Method	Latency ↓	Speed ↑	Score ↑
AI2D-lite	Vanilla	2.60	1.00×	43.75
	S^{2}I-dLLM†	1.02*	2.54×	42.71
ScienceQA-IMG	Vanilla	2.53	1.00×	68.17
	S^{2}I-dLLM	1.06	2.39×	68.32
MMMU-val	Vanilla	2.53	1.00×	47.11
	S^{2}I-dLLM	1.35	1.87×	46.44

C.3 Task-Conditioned Design Controls

Table 8 uses fixed subsets of 48 GenEval prompts (eight per category), 32 DPG prompts, 96 AI2D-lite questions, and 128 GSM8K questions, with four seeds per image prompt. Panel (a) matches retained channel counts; shared scores pool per-task, per-layer RMS-normalized saliency equally. Panel (b) matches candidate-layer budgets and probe intervals, freezing the first probe's skip set in the fixed variant. Panel (c) fixes refresh coverage at $\eta = 0.90$ and reports measured latency rather than matching it. Panel (d) uses a strict confidence threshold of 0.80 and a cap of five commitments per call for both backends.

Table 8: Task-conditioned design controls on MMaDA-8B-MixCoT.

	T2I: GenEval		MMU: AI2D-lite		Text: GSM8K	
Design variant	Score ↑	Time (s) ↓	Acc. (%) ↑	Time (s) ↓	Acc. (%) ↑	Time (s) ↓
(a) Channel ranking						
Shared ranking	0.5104	16.18	54.17	2.67	50.78	4.87
Task-specific ranking	0.5521	16.16	57.29	2.64	56.25	4.94
(b) Layer selection						
Fixed skip set	0.5521	16.18	57.29	2.65	57.81	4.95
Adaptive skip set	0.5519	15.94	57.21	2.60	57.63	4.86
(c) Refresh granularity						
Token-wise	0.5208	16.11	n/a	n/a	n/a	n/a
2×2 blocks	0.5156	16.36	n/a	n/a	n/a	n/a
Token-wise $\rightarrow 2\times2$	0.5521	16.16	n/a	n/a	n/a	n/a
(d) Answer-state backend						
Confidence on both tasks	n/a	n/a	55.21	8.17	54.69	5.16
ReMix on both tasks	n/a	n/a	54.17	8.25	53.91	5.16

Task-specific ranking scores higher than shared ranking on all three subsets, and depth-dependent refresh outperforms either uniform granularity on GenEval. Adaptive skip sets slightly reduce latency with small score decreases relative to fixed sets on all three subsets. Under matched backend thresholds and caps, confidence commitment slightly exceeds ReMix on both language tasks. The controls therefore support some design choices but not a universal advantage for adaptive selection or the MMU refinement backend.

C.4 Parameter Sensitivity and Quality–Latency Trade-offs

The initialization and refresh comparisons use a frozen development implementation of MMaDA-8B-MixCoT, with 24 matched GenEval prompt–seed pairs and 50 steps at 512×512. The reference uses 10% nominal FFN pruning, one candidate skipped layer, five initial full passes, $\beta = 0.30$, $\eta = 0.90$, refresh intervals of four, and a probe interval of ten.

Initialization and refresh intervals. Table 9 varies the initial full passes and image/condition refresh intervals. Reducing initialization from five to three passes preserves 17/24 passes while changing latency from 20.06 to 19.82 seconds. Longer refresh intervals do not consistently improve the quality–latency trade-off: probes, recovery, and execution-mode changes can trigger earlier refreshes, so nominal intervals need not translate into monotonic savings.

Independent recovery validation. Table 10 uses 24 other prompts with two new seeds each. The preselected $\beta = 0.10$ candidate reaches $1.38\times$ speedup but loses three passes relative to Vanilla, exceeding the prespecified tolerance of two out of 48. The reference $\beta = 0.30$ loses one pass at $1.21\times$ speedup. We retain this reference for the archived configuration; $\beta = 0.20$ was a diagnostic comparison rather than a preselected candidate.

Denoising schedule. Separately, Table 11 compares a fixed 35-call schedule with an adaptive-call variant on the full GenEval and DPG-Bench sets. Both retain the same structural and token

Table 9: Initialization and refresh sensitivity. Each row changes only the indicated setting.

Setting	Value	Passed / 24 ↑	Latency (s) ↓
Reference	–	17	20.06
Initial full passes	3	17	19.82
	4	15	19.61
Image refresh interval	5	15	19.54
	6	16	19.89
Condition refresh interval	6	12	19.85
	8	15	20.11
	10	13	19.51

Table 10: Recovery validation on held-out prompts and seeds.

Configuration	Passed / 48 ↑	Latency (s) ↓	Speedup ↑
Vanilla	22	24.39	1.00×
S^{2}I-dLLM ($\beta = 0.30$)	21	20.15	1.21×
S^{2}I-dLLM ($\beta = 0.20$)	20	18.81	1.30×
S^{2}I-dLLM ($\beta = 0.10$)	19	17.72	1.38×

mechanisms, calibration, and initial/final full-computation phases. The fixed schedule selects 35 calls from the original 50-step grid; the adaptive variant uses a prediction-margin threshold of 0.05 and an additional commitment fraction of 0.25. Matched prompt–seed pairs run on the same GPU, covering 2,212 GenEval and 4,260 DPG images per variant; ImageReward uses the DPG images. The comparison reports quality, NFE, and latency jointly to examine the effect of changing the denoising schedule.

Table 11: Fixed and adaptive denoising schedules with matched computation mechanisms.

Benchmark	Schedule	Score ↑	IR ↑	NFE ↓	Latency (s) ↓
GenEval	Fixed	0.5281	n/a	35.00	10.215
	Adaptive	0.5269	n/a	32.62	9.53
DPG-Bench	Fixed	71.30	0.2237	35.00	10.209
	Adaptive	71.18	0.2215	32.48	9.47

Evaluation scope. The development comparisons and independent validation are separate from the full-benchmark schedule comparison. Their quality metric is a local GenEval pass count rather than the full benchmark's category macro-average. Small samples and cross-batch timing limit fine-grained ranking. Schedule-comparison latencies use only its own same-GPU pairs, without mixing timings from other runs.

C.5 Additional Analysis of Computational Mechanisms

These diagnostics use existing unaccelerated MMaDA-8B-Base traces, with eight inputs per task. The legacy T2I path ignores the padding mask; MMU/Text inputs have no padding. Feature drift and fixed-state bypass perturbations characterize local behavior, not end-task quality or validated acceleration. Repeated layer/state observations are not independent samples; marked regions are descriptive rather than execution thresholds.

Main-figure measurement details. Figure 1 averages eight trajectories per task at 25%, 50%, and 75% denoising progress. Channel importance is $\Delta_j = \mathbb{E}[h_j^2]\|W^o_{:,j}\|_2^2$ over 12,288 channels. This diagnostic score summarizes channel energy and is distinct from the activation–weight saliency used for channel selection in the method. Inter-layer cosine similarity follows Ke et al. (2026), with a shared $[0.9, 1]$ color range and lower values clipped; colored annotations mark neighboring-layer similarity and distant-layer differences, while ellipses are descriptive. Figure 3 measures adjacent-step V drift, $D = 1 - \cos(V_t, V_{t-1})$, averaged over eight inputs. Role maps span layers 1–32 horizontally and native transitions vertically, using shared square-root colors and task-specific ranges

for temporal-mean bars. Spatial maps show the first T2I input at its middle captured state on a shared linear scale; Moran's I compares observed and shuffled positions over eight inputs (mean $\pm$ sample SD).

C.5.1 Layer Bypass Sensitivity

Figure 6 pairs layer change with the prediction KL induced by identity bypass at the same state. Each task contains eight inputs, three captured phases, and all network layers. Spearman correlations summarize the association; layer seven is highlighted. T2I uses conditional image-code logits without CFG, whereas MMU/Text use the full vocabulary.

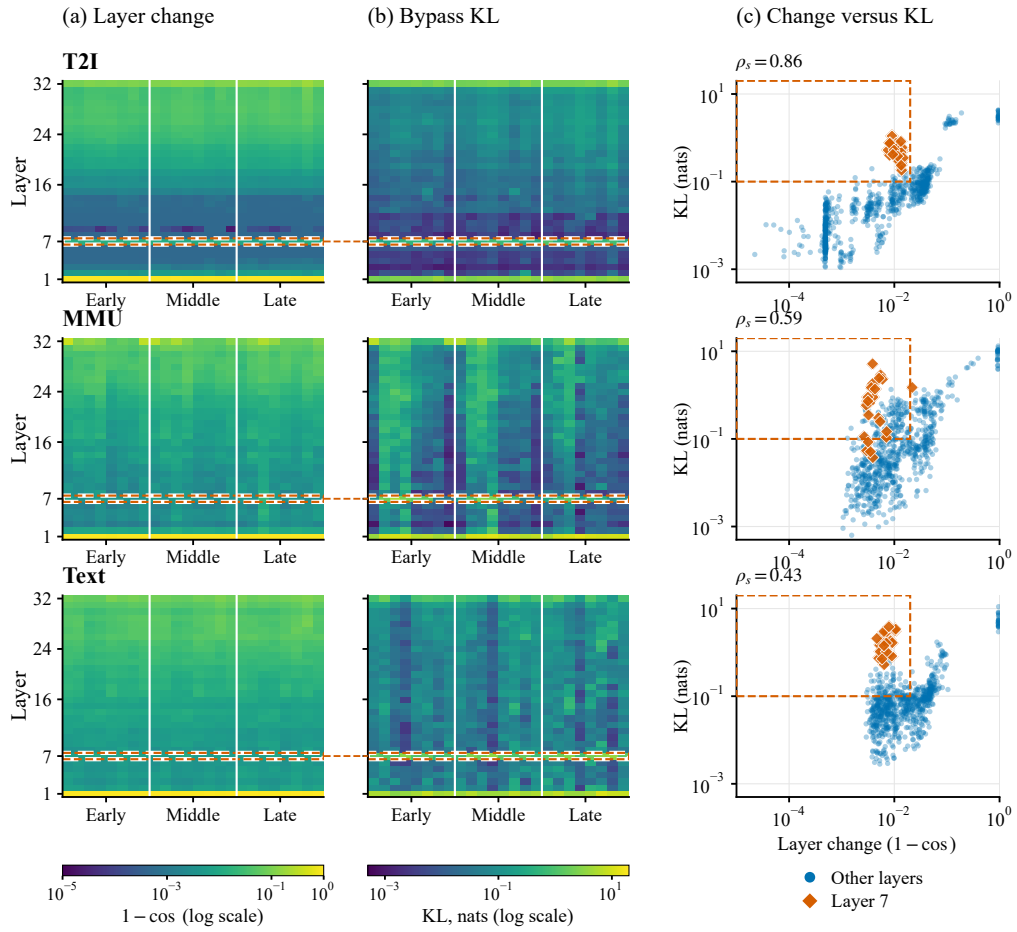


Figure 6: Layer bypass sensitivity. (a) Layer change. (b) Prediction KL after bypassing one layer. (c) Their relationship at matched states; diamonds highlight the selected layer.

C.5.2 Layer and Token State Analysis

Figure 7 combines regional bypass analysis and token trajectories. Regions partition the same bypass observations at angular change 0.02 and KL 0.1; boxplots show medians, interquartile ranges, and 5th–95th percentile whiskers. These are descriptive partitions, not execution thresholds. Token trajectories pair target V drift D with the revised-target fraction R, including committed and EOS positions. Thin lines show individual inputs and bold lines their equal-weight mean; shaded hulls summarize temporal thirds. Shared log1p-transformed axes retain raw-value ticks. These traces use the archived unaccelerated schedules, with 18 T2I transitions and 256 MMU/Text transitions under 32-token blocks.

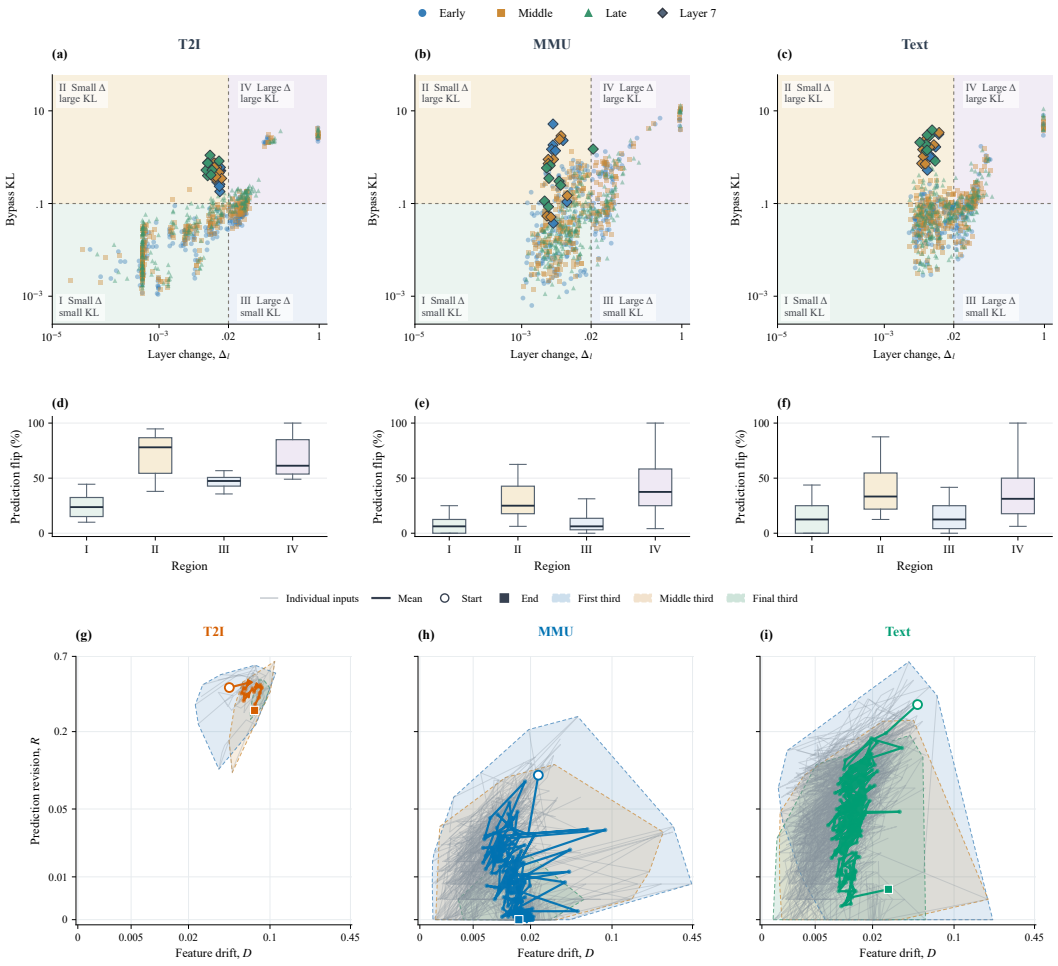


Figure 7: Layer and token diagnostics. (a–c) Layer change versus bypass KL. (d–f) Prediction flip rates by region. (g–i) Feature drift versus prediction revision along denoising trajectories. Columns correspond to T2I, MMU, and Text.

D Computational and Memory Cost Analysis

Let B, N, D, and H denote the effective batch size, sequence length, hidden dimension, and attention head count. We consider standard multi-head attention with equal Q/K/V widths, count a multiply-add as two FLOPs, and approximate softmax by five operations per attention entry. Batched classifier-free guidance branches are included in B.

Dense computation. For a full sequence, self-attention requires

$$\begin{aligned} \mathcal{C}_{\mathrm{SA}} &\approx \underbrace{6BND^2}_{\mathrm{QKV}} + \underbrace{2BN^2D}_{QK^\top} + \underbrace{5BHN^2}_{\mathrm{Softmax}} + \underbrace{2BN^2D}_{\mathrm{Attn}\times V} + \underbrace{2BND^2}_{\mathrm{OutProj}} \\ &= 8BND^2 + 4BN^2D + 5BHN^2. \end{aligned} \tag{15}$$

Using the full FFN width d_{ff} defined in Section 3.5, the three SwiGLU projections contribute

$$\mathcal{C}_{\mathrm{FFN}} \approx \underbrace{2BNDd_{\mathrm{ff}}}_{\mathrm{Gate}} + \underbrace{2BNDd_{\mathrm{ff}}}_{\mathrm{Up}} + \underbrace{2BNDd_{\mathrm{ff}}}_{\mathrm{Down}} = 6BNDd_{\mathrm{ff}}. \tag{16}$$

Thus, the leading cost per block is $\mathcal{C}_{\mathrm{block}} \approx \mathcal{C}_{\mathrm{SA}} + \mathcal{C}_{\mathrm{FFN}}$. Normalization, rotary embeddings, residual additions, and elementwise activations are accounted for separately below.

Scheduled computation. Following Appendix B.1, f indexes actual backbone calls. At layer l, let $r_{l,f}$ count positions recomputed by Q/K, attention output, and FFN, and let $v_{l,f}$ count positions with newly projected V. The attention context has length N_f, including cached positions. Let p_f denote the generation progress at call f. With the executed width $d_{\tau,l}(p_f)$ from Equation (8), the leading cost of an executed block is

$$\begin{aligned}\widetilde{\mathcal{C}}_{l,f} \approx\ & \underbrace{6Br_{l,f}D^2}_{\text{Q/K/OutProj}} + \underbrace{2Bv_{l,f}D^2}_{\text{V}} \\ & + \underbrace{4Br_{l,f}N_fD + 5BHr_{l,f}N_f}_{\text{Attention}} + \underbrace{6Br_{l,f}D\,d_{\tau,l}(p_f)}_{\text{SwiGLU}}.\end{aligned} \tag{17}$$

The counts refer to actually recomputed positions, including any refreshed conditioning positions. Attention retains the full key/value context and therefore scales with $r_{l,f}N_f$, rather than $r_{l,f}^2$.

Relative to the dense FFN, token refresh and width selection combine as

$$\frac{\widetilde{\mathcal{C}}_{\mathrm{FFN},l,f}}{\mathcal{C}_{\mathrm{FFN},f}} \approx \frac{r_{l,f}}{N_f}\,\frac{d_{\tau,l}(p_f)}{d_{\mathrm{ff}}}. \tag{18}$$

Here $\mathcal{C}_{\mathrm{FFN},f}$ is the dense cost for length N_f. This relation makes the joint effect of token scheduling and computational sparsification explicit.

Denoising cost. Let F_τ be the actual number of backbone calls and L the number of layers. Reusing the layer gate $m_{l,f}$ from the main text, the denoising cost is

$$\mathcal{C}^{(\tau)}_{\mathrm{denoise}} \approx \sum_{f=1}^{F_\tau}\sum_{l=1}^{L} m_{l,f}\widetilde{\mathcal{C}}_{l,f} + \mathcal{C}_{\mathrm{head}} + \mathcal{C}_{\mathrm{aux}}. \tag{19}$$

The output projection contributes $\mathcal{C}_{\mathrm{head}} \approx \sum_f 2Bn_{\mathrm{head},f}D|\mathcal{V}_\tau|$, where $n_{\mathrm{head},f}$ counts projected positions and $|\mathcal{V}_\tau|$ is the output vocabulary size. $\mathcal{C}_{\mathrm{aux}}$ includes normalization, elementwise operations, refresh scoring, and prediction-state updates. Layer bypass sets $m_{l,f} = 0$; full probes set all gates to one, restore d_{ff}, and use $r_{l,f} = v_{l,f} = N_f$. Recovery and cache rebuilding enter through their actual executed widths and refreshed positions, while generation progress determines F_τ.

Task-dependent computational cost. Text-to-image. Let N_c and N_I denote text-condition and image-target lengths, and let $c_{l,f} \in \{0,1\}$ indicate whether the condition is refreshed. With the refresh set $\mathcal{R}_{l,f}$ from Equation (10),

$$N_f = N_c + N_I, \qquad r_{l,f} = |\mathcal{R}_{l,f}| + c_{l,f}N_c, \qquad v_{l,f} = N_I + c_{l,f}N_c. \tag{20}$$

All image V projections are evaluated for refresh scoring, while Q/K, attention outputs, and FFN outputs are updated only at selected image positions. Spatial groups contribute their constituent token counts to $|\mathcal{R}_{l,f}|$. Savings therefore arise within each call through selective refresh, width selection, and layer bypass; the call count remains explicit in Equation (19).

Multimodal understanding. Here N_c includes the image and question prefix. Let $N_{A,f}$ denote the answer canvas recomputed by the decoder, including positions outside the current commitment set. For prefix reuse with per-pass answer recomputation,

$$N_f = N_c + N_{A,f}, \qquad r_{l,f} = v_{l,f} = N_{A,f} + c_{l,f}N_c. \tag{21}$$

Prefix rebuilding sets $c_{l,f} = 1$. Continuous refinement changes F_{MMU}, while candidate mixing and prediction-history comparisons contribute to $\mathcal{C}_{\mathrm{aux}}$. The recomputed answer length is not replaced by the number of uncommitted positions.

Text reasoning. Equation (21) also applies, with a text-only prefix. Confidence-gated parallel commitment changes F_{Text}; confidence evaluation and position selection enter $\mathcal{C}_{\mathrm{aux}}$. Committed positions within the recomputed canvas still contribute to the per-call cost. For either language task, enabled structural reduction enters through $d_{\tau,l}(p_f)$ and $m_{l,f}$; a full-width, full-depth configuration uses d_{ff} and unit gates.

For complete requests, condition encoding and output decoding add to $\mathcal{C}^{(\tau)}_{\mathrm{denoise}}$: MMU includes visual encoding, and T2I includes image decoding. FLOPs reduction does not directly specify latency, which also depends on cache traffic and kernel execution.

Memory cost. Let s_w and s_a denote bytes per weight and activation element, respectively, and P the number of resident model parameters. Weight storage is $M_{\text{weights}} = s_w P$. S^{2}I-dLLM reorders the full FFN weights and executes a selected prefix; retaining the remaining channels for recovery leaves weight storage unchanged. Layer bypass likewise reduces execution rather than parameter storage.

For compact cache buffers, let $\mathcal{L}_c$ be the set of layers retaining caches and n_l^z the number of stored positions for feature type z. Under the same attention-width assumption as above, the feature-cache payload is

$$M_{\text{cache}} \approx s_a BD \sum_{l \in \mathcal{L}_c} \left(n_l^K + n_l^V + n_l^A + n_l^F \right), \tag{22}$$

where A and F denote attention and FFN outputs, both of width D. Caching all four features over N positions gives $M_{\text{cache}} \approx 4 s_a BDN|\mathcal{L}_c|$. Actual storage counts shared backing buffers once and includes any retained capacity beyond the logical views. Refreshing fewer positions reduces updates, but does not proportionally shrink the retained cache; skipping a layer also does not free its cache unless that buffer is released.

For an executed FFN, the intermediate activation storage scales as

$$M_{\text{FFN,temp},l,f} = O(s_a B r_{l,f} d_{\tau,l}(p_f)). \tag{23}$$

Thus, token selection and width reduction jointly reduce this component. To account for coexisting buffers, let λ index execution instants, M_{state} cover persistent decoding states and scheduling metadata, and M_{temp} cover temporary activations, logits, and operator workspaces. Peak tensor storage is

$$M_{\text{peak}} = M_{\text{weights}} + \max_{\lambda} \left[M_{\text{cache}}(\lambda) + M_{\text{state}}(\lambda) + M_{\text{temp}}(\lambda) \right]. \tag{24}$$

The terms count disjoint live storage, excluding unused allocator reservations. Full probes, width recovery, and cache rebuilding can increase temporary storage, so reduced FLOPs alone do not imply a lower memory peak.

Task-dependent memory cost. T2I retains image features across calls, so its cache size depends on stored image positions rather than the refresh-set size. MMU and text reasoning retain prefix features; any answer buffers retained by the implementation are also included in Equation (22). Their distinct decoding states contribute differently to M_{state}.

For MMU, let N_{hist} be the number of positions retained in the dense prediction-history tensor and s_p its bytes per probability. The history contributes

$$M_{\text{hist}}^{\text{MMU}} = s_p B N_{\text{hist}} |\mathcal{V}_{\text{MMU}}|. \tag{25}$$

The current decoder retains the full predicted distribution rather than only the candidates used for mixing. Continuous embeddings and candidate gathering also contribute to temporary storage. Text reasoning does not maintain this refinement history; its token and commitment states scale with sequence length, while current logits and probabilities still occupy temporary memory. For all three tasks, cache reconstruction and full-computation recovery remain included in the peak-storage accounting above.

E Qualitative Results and Limitations

E.1 Qualitative Outputs and Generation Process Examples

Figure 8 presents selected DPG-Bench outputs from the 50-step S^{2}I-dLLM reference configuration, covering varied objects, attributes, and scene compositions. We use the first generated sample for each selected prompt and retain its complete original wording.

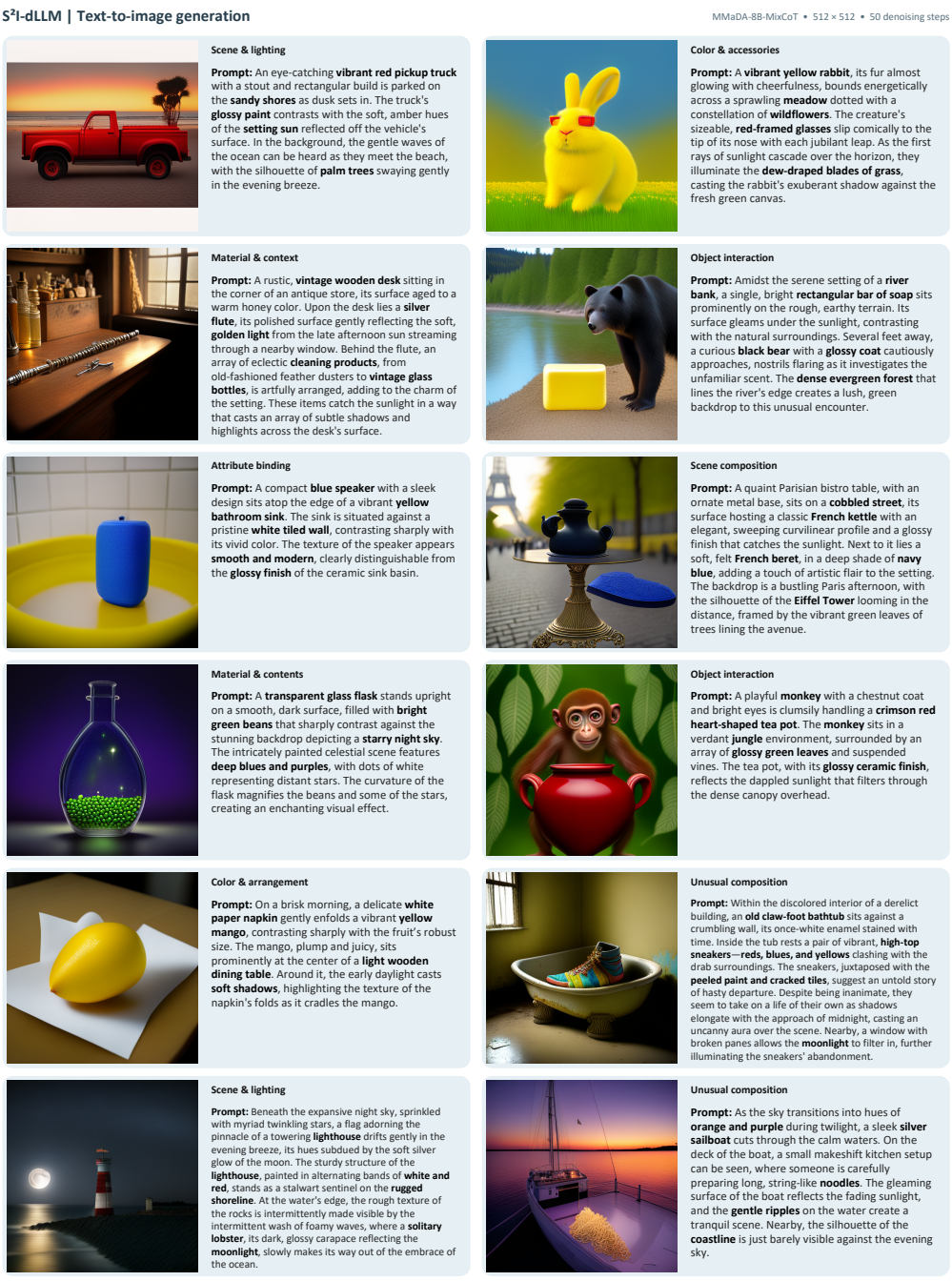


Figure 8: Text-to-image examples from S^{2}I-dLLM on DPG-Bench. Bold text highlights prompt attributes.

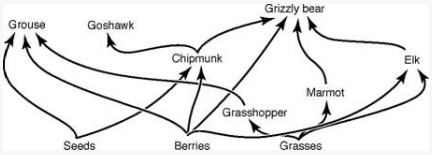
E.1.1 Multimodal Understanding Case Study

Figure 9 presents an AI2D example selected from the frozen joint configuration with both channel sparsification and layer skipping active. The original 256-step baseline and the accelerated configuration both return the reference answer, C (Producer), with recorded latencies of 47.09 s and 3.65 s, respectively. The same-run ReMix control uses the same 26 forward calls as the joint configuration but takes 4.82 s, illustrating a reduction in per-pass cost with unchanged NFE. The baseline–S^{2}I-dLLM latency ratio is a cross-batch, single-example comparison across code revisions; both runs

share model weights, input tokens, seed, generation budgets, and GPU type. Timing includes preprocessing.

Case: AI2D Generation length: 256 Block length: 128

Input image



Question: Berries are classified as what?

A. Consumer
B. Decomposer
C. Producer
D. None of the above

Ground truth: C. Producer

Recorded execution

	Forward calls	Active E2E	Answer
Baseline (MMaDA-8B-MixCoT)	256	47.09 s	C
ReMix control	26	4.82 s	C
S^{2}I-dLLM (joint + prefix cache)	26	3.65 s	C

12.89× observed overall speedup (cross-batch)

Baseline: final response

<think>
To determine what berries are classified as, we need to understand the food chain provided. In this diagram, berries are shown as a consumer that is eaten by worms. This means berries are classified as a producer.

Therefore, the correct answer is:
C
</think>
C

S^{2}I-dLLM: final response

<think>
To determine what berries are classified as, let's analyze the food chain provided:

1. Berries are the starting point of the chain.
2. Berries are connected to other organisms.
3. This means that berries are producing organisms in the food chain, providing energy to other organisms.

Therefore, berries are classified as producers.
</think>
C

Actual structural execution

15 of 26 forward calls use the compact FFN width (9,792 of 12,288 channels); 15 layer calls are skipped in total. The trace also records one event-triggered full recovery, two periodic probes, and five final-phase full-computation calls.

Figure 9: Correct multimodal understanding with structural sparsification and token scheduling. Responses are reproduced verbatim.

E.1.2 Text Reasoning Case Study

Figure 10 compares paired final outputs from the GSM8K evaluation in Table 2, using the same input and seed with a generation length of 256 and a block length of 64. Both methods correctly compute the total number of baby outfits as 87, with different wording in their generated reasoning. S^{2}I-dLLM uses 80 rather than 256 forward calls, with a recorded request latency of 3.01 s versus 14.67 s. This example uses the token-only configuration; the recorded block-completion counts illustrate its variable decoding progress.

Case: GSM8K Generation length: 256 Block length: 64

Question: Laurel's friend gave her 24 baby outfits that her child no longer needed. At her baby shower, Laurel received twice the amount of new baby outfits. Then, Laurel's mom gifted her with another 15 baby outfits. How many outfits does she have for her baby?

Ground truth: 87

Baseline (MMaDA-8B-MixCoT) NFE: 256 Latency: 14.67 s

Final generated reasoning and answer (correct)

First, we need to determine how many new baby outfits Laurel received at her baby shower. Since she received twice the amount of new baby outfits as her friend gave her 24, **we multiply 24 by 2 to find the number of new baby outfits.**

Next, we add the number of new baby outfits to the original 24 baby outfits to find the total number of baby outfits she has. This gives us 24 + 48 = 72 outfits.

Finally, we add the number of baby outfits that Laurel's mom gifted her, which is 15, to the total number of baby outfits she has. This gives us 72 + 15 = 87 outfits.

</think>

$\boxed{87}$

S^{2}I-dLLM (token) NFE: 80 Latency: 3.01 s

Final generated reasoning and answer (correct)

First, let's calculate the number of new baby outfits Laurel received at her baby shower. She received twice the amount of her friend's, which was 24. **So, 2 * 24 = 48 new baby.**

Next, let's add the number of new baby outfits to the original 24 that her friend gave her. 24 + 48 = 72.

Finally, let's add the number of baby outfits Laurel's mom gave her, which is 15. 72 + 15 = 87.

</think>

$\boxed{87}$

Recorded S^{2}I-dLLM decoding progress. The four 64-position blocks complete after 44, 77, 79, and 80 forward calls. These counts cover the full generation buffer, including end-of-sequence/padding positions.

Figure 10: Paired correct reasoning on GSM8K. S^{2}I-dLLM uses fewer forward calls; key calculations are highlighted.

E.2 Failure Cases and Scope of Applicability

Figure 11 shows a paired failure case: the baseline accounts for the different working hours correctly, whereas the accelerated output misassigns the hours and returns an incorrect answer. Fewer forward calls therefore do not guarantee preserved correctness on every input, consistent with the aggregate GSM8K accuracy trade-off in Table 2. The saved records contain final responses and submission counts, but no intermediate response snapshots; they do not establish when the reasoning error first appeared.

Case: GSM8K Generation length: 256 Block length: 64

Question: Tabitha agreed to pay John and Jill $10 an hour to help clean out her attic and basement. Jill worked 2 hours on Saturday and 1 hour on Sunday. John worked twice as long as Jill on Saturday and three times as long as Jill on Sunday. How much more money did John earn compared to Jill?

Ground truth: 40

Baseline (MMaDA-8B-MixCoT) NFE: 256 Latency: 14.98 s

Final generated reasoning and answer (correct)

First, let's calculate the total number of hours Jill worked. She worked 2 hours on Saturday and 1 hour on Sunday, so she worked 2 + 1 = 3 hours in total.

GSM8K failure case (continued)

Next, let's calculate the total number of hours John worked. He worked twice as long as Jill on Saturday, which is 2 * 2 = 4 hours. He also worked three times as long as Jill on Sunday, which is 3 * 1 = 3 hours. So, John worked 4 + 3 = 7 hours in total.

Now, let's calculate the total earnings for each person. Jill earned $10 per hour for 3 hours, so she earned 3 * $10 = $30. John earned $10 per hour for 7 hours, so he earned 7 * $10 = $70.

Finally, to find out how much more money John earned compared to Jill, we subtract Jill's earnings from John's earnings: $70 - $30 = $40.

</think>

40

S^{2}I-dLLM (token) NFE: 87 Latency: 3.07 s

Final generated reasoning and answer (incorrect)

First, let's calculate how much John earned on Saturday. **He worked 2 hours, so he earned 2 * $10 = $20.**

Next, let's calculate how much John earned on Sunday. He worked 1 hour, so he earned 1 * $10 = $10.

Now, let's calculate how much John earned in total. He earned $20 on Saturday and $10 on Sunday, so he earned $20 + $10 = $30.

Finally, let's calculate how much John earned compared to Jill. Jill earned $20 on Saturday and $10 on Sunday, so John earned $20 + $10 = $30 more than Jill.

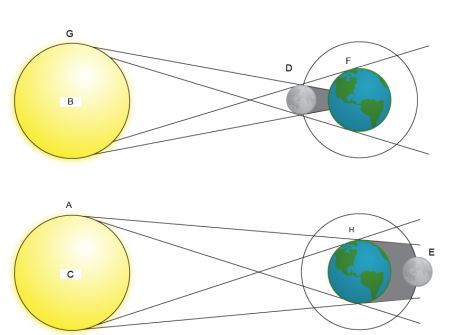
</think>

0

Figure 11: GSM8K failure case. S^{2}I-dLLM misassigns the working hours, producing an incorrect answer.

Multimodal understanding failure. Figure 12 shows an AI2D lunar-eclipse example from the archived joint diagnostic evaluation, with channel reduction and layer skipping active. The baseline's final option matches the reference, C (diagram label E), while S^{2}I-dLLM selects B and confuses the Sun–Earth–Moon arrangement. The continuous-refinement control also selects B, so this example does not isolate structural reduction as the source of the error. The card reproduces excerpts of final responses; intermediate decoding snapshots are unavailable.

Case: AI2D Multimodal understanding failure



Question: Which label indicates the moon during a lunar eclipse?

A. D B. C C. E D. B

Ground truth: C (diagram label E)

Recorded outputs

Method	Forward calls	Answer
Baseline	256	C
Continuous-refinement control	31	B
S^{2}I-dLLM (joint diagnostic configuration)	54	B

AI2D failure case (continued)

Baseline: final-response excerpt

Therefore, the correct answer is option E.
</think>
C

S^{2}I-dLLM: final-response excerpts

To determine which label indicates the moon during a lunar eclipse, we need to understand the positions of the Sun, Earth, and Moon in the diagram. During a lunar eclipse, the Moon passes between the Earth and the Sun, blocking the Earth's shadow on the Moon. This creates the positions of the Moon as the Moon's shadow through the Earth's shadow.

[...]

During the lunar eclipse, the Moon passes through the Earth's shadow. Therefore, the correct answer is: B
</think>

Execution configuration

Generation length 256; block length 128. Compact calls retain 9,792 of 12,288 FFN channels, with seven layer calls skipped over the trajectory. This diagnostic run uses continuous refinement without prefix feature caching.

Figure 12: A multimodal understanding failure. Verbatim excerpts of final responses; the baseline selects the reference option, while both accelerated configurations select an incorrect option.