

D³ToM: Decider-Guided Dynamic Token Merging for Accelerating Diffusion MLLMs

Shuochen Chang, Xiaofeng Zhang*, Qingyang Liu, Li Niu[†]

MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{csc1332741686, framebreak, narumimaria, ustcnewly}@sjtu.edu.cn

Abstract

Diffusion-based multimodal large language models (Diffusion MLLMs) have recently demonstrated impressive non-autoregressive generative capabilities across vision-and-language tasks. However, Diffusion MLLMs exhibit substantially slower inference than autoregressive models: Each denoising step employs full bidirectional self-attention over the entire sequence, resulting in cubic decoding complexity that becomes computationally impractical with thousands of visual tokens. To address this challenge, we propose D³ToM, a Decider-guided dynamic token merging method that dynamically merges redundant visual tokens at different denoising steps to accelerate inference in Diffusion MLLMs. At each denoising step, D³ToM uses decider tokens—the tokens generated in the previous denoising step—to build an importance map over all visual tokens. Then it maintains a proportion of the most salient tokens and merges the remainder through similarity-based aggregation. This plug-and-play module integrates into a single transformer layer, physically shortening the visual token sequence for all subsequent layers without altering model parameters. Moreover, D³ToM employs a merge ratio that dynamically varies with each denoising step, aligns with the native decoding process of Diffusion MLLMs, achieving superior performance under equivalent computational budgets. Extensive experiments show that D³ToM accelerates inference while preserving competitive performance. The code is released at <https://github.com/bcml/D3ToM-Diffusion-MLLM>.

1 Introduction

Diffusion-based large language models (Diffusion LLMs) have recently emerged as a promising alternative generative paradigm to autoregressive LLMs (Yang et al. 2025a). They implement a discrete diffusion process at inference by starting from a sequence of mask tokens and iteratively decoding discrete text tokens via a trained reverse diffusion process. Pioneering Diffusion-LLMs such as LLaDA (Nie et al. 2025; Zhu et al. 2025) and Dream (Ye et al. 2025a), have achieved comparable results to autoregressive LLMs across diverse language tasks (Ye et al. 2024; Gong et al. 2025a,b; Wen et al. 2025; Huang et al. 2025b). Subsequently, LLaDA-V (You et al. 2025), MMaDA (Yang et al. 2025c) and

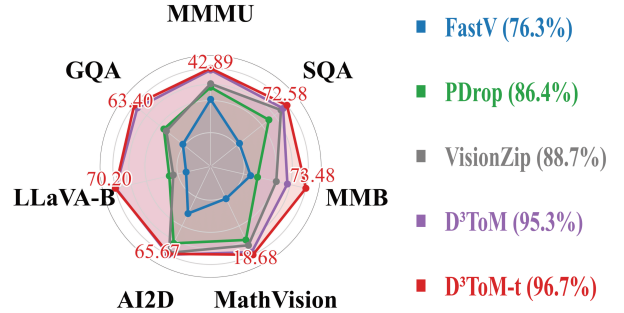


Figure 1: **D³ToM Performance.** Our D³ToM outperforms the current SOTA methods such as FastV, PDrop and VisionZip, achieving over 96% of the performance with only 10% of the visual tokens on LaViDa.

LaViDa (Li et al. 2025) extend this paradigm to vision-and-language tasks via a visual instruction tuning framework that integrates a vision encoder (Tschannen et al. 2025) and connector to map visual features into the Diffusion LLM’s text embedding space. Diffusion MLLMs achieve impressive non-autoregressive generative capabilities across multimodal benchmarks.

Diffusion MLLMs theoretically allow faster decoding than autoregressive models due to their capability of parallel decoding multiple tokens per denoising step, unlike the autoregressive models’ sequential one-token-per-step decoding. However, in practice Diffusion MLLMs suffer from cubic time complexity during full-sequence decoding (Ma et al. 2025). Formally, given a total sequence length N (including input and output tokens) and $T = \mathcal{O}(N)$ denoising steps, each step involves full bidirectional self-attention over all N tokens with a complexity of $\mathcal{O}(N^2)$, thus resulting in an overall cubic complexity of $\mathcal{O}(N^3)$. Moreover, Diffusion MLLMs process thousands of visual tokens, making N far larger than in text-only settings and amplifying the cubic cost of full-sequence denoising. Recent advances in Diffusion LLMs, such as Fast-dLLM (Wu et al. 2025), dLLM-Cache (Liu et al. 2025b), dKV-Cache (Ma et al. 2025), and Prefix-DLM (Li et al. 2025), accelerate inference by integrating existing KV-Cache mechanisms into bidirectional attention. Parallel decoding schemes such as Confidence-Aware Parallel Decoding (Wu et al. 2025) and

*Project leader.

[†]Corresponding author.

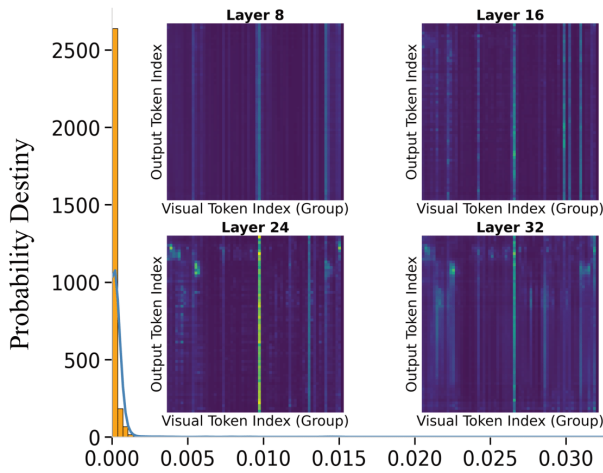


Figure 2: **Illustration of visual token redundancy.** Probability density distribution of attention weights from output tokens to grouped visual tokens. Because an image contains over one thousand visual tokens while the decoder generates only 64 output tokens, every 16 consecutive visual tokens are averaged to form a single group, and attention from each output token is aggregated over these groups. The resulting distribution shows that most visual-token groups receive near-zero attention, indicating substantial redundancy.

SlowFast (Wei et al. 2025) enable simultaneous generation of multiple tokens per denoising step. However, these strategies do not reduce the number of visual tokens, so N remains large and the decoding cost persists in diffusion MLLMs.

In our study, we find that Diffusion MLLMs exhibit considerable redundancy among visual tokens. As shown in Figure 2, the self-attention weight distribution is heavily skewed toward zero and attention values concentrate on a small subset of visual tokens, indicating that only a few tokens capture the majority of visual information while the rest contribute minimally. Although autoregressive MLLM research has leveraged this redundancy through token pruning and merging strategies (Bolya et al. 2023; Feng et al. 2024; Choi et al. 2024; Hyun et al. 2025; Chen et al. 2024; Zhang et al. 2024b; Jeddi et al. 2025; Yang et al. 2025d; Zhang et al. 2024a), similar token compression techniques for Diffusion MLLMs remain unexplored. Furthermore, we observe that the subset of visual tokens receiving high attention shifts significantly across successive denoising steps. As illustrated in Figure 3, early iterations concentrate on the Chihuahua’s body, corresponding to the decoding of “white”, whereas later steps focus attention on the background stump as the word “stump” is generated. This evolving attention pattern reveals a step-wise sparsity in visual saliency that static pruning cannot effectively capture.

Motivated by these dynamic attention patterns, we introduce D³ToM, a decider-guided dynamic token merging method for diffusion MLLMs. We define decider tokens as the output token decoded at the previous denoising iteration. At each denoising step t , D³ToM treats the attention weights

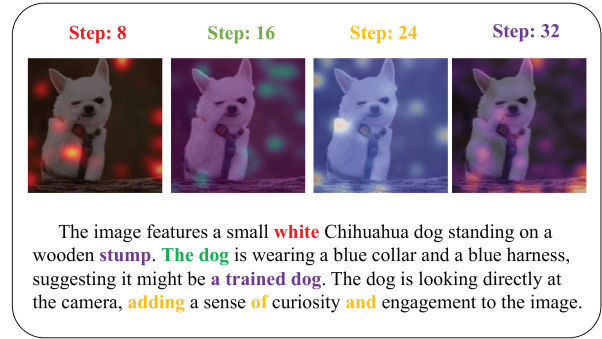


Figure 3: **Visualization of attention weights.** The figure displays four snapshots of attention weights at different denoising steps (8, 16, 24, and 32) assigned to visual tokens. Each snapshot shows attention distribution of the output tokens generated at the current denoising step on the input image of a small white Chihuahua dog on a wooden stump. The highlighted regions in each image represent the areas of the input image that receive the most attention from the output tokens generated at that specific decoding step. Different colors indicate the attention focus of output tokens generated at different steps.

from these decider tokens to all visual tokens as importance scores. Given a merge ratio α , it selects the top $1 - \alpha$ fraction of visual tokens to keep. For each of the remaining tokens, the method computes its similarity to all kept tokens and merges it into the most similar kept token by aggregating their embedding vectors. This merging is executed within one transformer layer, physically reducing the visual sequence for all downstream layers without modifying any model parameters, and thereby accelerating the denoising inference process.

Moreover, D³ToM modulates its merge ratio α across denoising timesteps to match the shift from global semantic construction to fine-grained refinement. Initial iterations demand broad visual coverage and thus retain more tokens, while later iterations operate on increasingly localized details and can sustain stronger merging. This timestep-dependent adjustment ensures that token merging intensity adapts to dynamic shifts in visual saliency during the diffusion process.

Our extensive experiments validate the proposed design from three key perspectives. For performance, our method retains over 96% of the baseline model’s accuracy across seven standard vision-language benchmarks, even when compressing the visual sequence to just 10% of its original size (Sec.4.2). In terms of efficiency, this same configuration reduces computational cost to only 30% of the baseline FLOPs and 43% of the wall-clock time, matching the performance of state-of-the-art inner-LLM pruning techniques (Sec.4.3). Finally, our method demonstrates compatibility with other optimizations (Sec.4.4). When combined with KV-Cache, D³ToM yields complementary savings, confirming that our token merging approach addresses an orthogonal computational bottleneck.

Our contributions are summarized as follows:

- We propose D³ToM, a novel token merging strategy guided by decider tokens. Our method significantly reduces sequence length by exploiting visual redundancy and adaptively adjusts its merge ratio throughout the denoising process to maintain high model performance.
- Our method is designed as a training-free, plug-and-play module requiring no modification of existing model parameters. This ensures broad applicability and seamless integration into diverse Diffusion MLLM architectures.
- We demonstrate that our approach is orthogonal to existing KV caching optimizations. When used in conjunction, these methods yield additive efficiency gains by addressing different computational bottlenecks.

2 Related Work

2.1 Visual Token Reduction in Autoregressive MLLMs

To mitigate the computational overhead of lengthy visual token sequences in autoregressive MLLMs, several methods (Xu et al. 2025a; Huang et al. 2025a; Xu et al. 2025b; Liu et al. 2025a; Jiang et al. 2025; Ye et al. 2025b; Alvar et al. 2025; Yang et al. 2025b; Kallini et al. 2025) have been proposed. For instance, FastV (Chen et al. 2024) prunes tokens with minimal attention scores at intermediate layers. LLaVA-PruMerge (Shang et al. 2024) adaptively prunes less informative tokens via attention sparsity and subsequently merges them using a k-nearest neighbor strategy. Pyramid-Drop (Xing et al. 2024) progressively reduces token counts in stages with a lightweight attention-based strategy. Other approaches focus on different criteria. SparseVLM (Zhang et al. 2024b) prunes visual tokens with low contribution to text-related signals and reconstructs compact representations through clustering. VisionZip (Yang et al. 2025d) merges dominant visual tokens based on attention and similarity scores. HoliTom (Shao et al. 2025) employs pruning by spatial-temporal merging for video LLMs. However, these effective approaches are fundamentally designed for autoregressive specific architectures, rendering them incompatible with the non-causal denoising process in diffusion based MLLMs.

2.2 Efficiency Enhancements for Diffusion LLMs

Diffusion LLMs have primarily pursued two directions. The first adapts caching mechanisms to the non-causal attention of diffusion models. Methods like Fast-dLLM (Wu et al. 2025), dKV-Cache (Ma et al. 2025), dLLM-Cache (Liu et al. 2025b), and Prefix-DLM (Li et al. 2025) introduce tailored strategies to cache key-value pairs from stable prefixes or employ adaptive schedules, reducing redundant computations across denoising steps. The second direction focuses on accelerated sampling. Techniques such as Confidence-Aware Parallel Decoding (Wu et al. 2025), Confident Decoding in Dimple (Yu, Ma, and Wang 2025), and Slow-Fast (Wei et al. 2025) Sampling reduce the required denoising iterations by dynamically reordering or parallelizing token decoding. Although these advances substantially accelerate Diffusion LLMs, they do not specifically address the

redundancy and computational burden associated with extensive visual token sequences in Diffusion MLLMs. Consequently, the central challenge posed by long visual sequences remains unresolved.

3 Method

3.1 Preliminary: Diffusion MLLMs

Diffusion MLLMs are a class of generative models that produce text conditioned on multimodal inputs through a non-autoregressive denoising process. The core mechanism involves progressively refining a fully masked sequence into a coherent output that is contextually grounded in both visual and textual information.

Let [MASK] be a dedicated mask token in the token vocabulary set. The model receives a visual input, such as an image I , and a text prompt P . A vision encoder first processes the image to produce a sequence of visual embeddings $V = E_{\text{vision}}(I)$. The total sequence length for the model’s forward pass is denoted by N , which includes visual tokens, prompt tokens, and the text sequence to be generated.

The model generates a response sequence $X = (x_1, \dots, x_O)$ of a fixed length $|O|$ over T discrete denoising steps, indexed by $t = T, \dots, 1$. Let $X^{(t)}$ denote the state of the generated text sequence at step t . The process begins with a sequence composed entirely of mask tokens:

$$X^{(T)} = ([\text{MASK}], \dots, [\text{MASK}]). \quad (1)$$

At each step t , a Transformer-based model p_θ with L_{model} layers, parameterized by θ , predicts the probability distribution over the clean sequence X given the current noisy state $X^{(t)}$ and the multimodal context (V, P) :

$$p_\theta(X|X^{(t)}, V, P). \quad (2)$$

From this distribution, the most probable clean sequence, $\hat{X}$, is typically identified via greedy decoding. Subsequently, a scheduling function $\mathcal{S}$ determines the state for the next step, $X^{(t-1)}$, by selectively replacing a subset of mask tokens in $X^{(t)}$ with the corresponding predicted tokens from $\hat{X}$:

$$X^{(t-1)} = \mathcal{S}(\hat{X}, X^{(t)}, t). \quad (3)$$

This iterative process is repeated until $t = 1$, yielding the final generated sequence $X^{(0)}$. The set of newly predicted tokens in the transition from $X^{(t)}$ to $X^{(t-1)}$ will serve as crucial guides in the subsequent denoising step, as we will detail next.

3.2 Decider-Guided Dynamic Token Merging

The core of our methodology is D³ToM, a decider-guided dynamic token merging strategy. We design it as a plug-and-play module to address the computational demands of the iterative denoising process. For clarity, we denote the matrix of hidden states for a sequence of length N as $\mathcal{H} \in \mathbb{R}^{N \times d_{\text{model}}}$, where d_{model} is the hidden dimension.

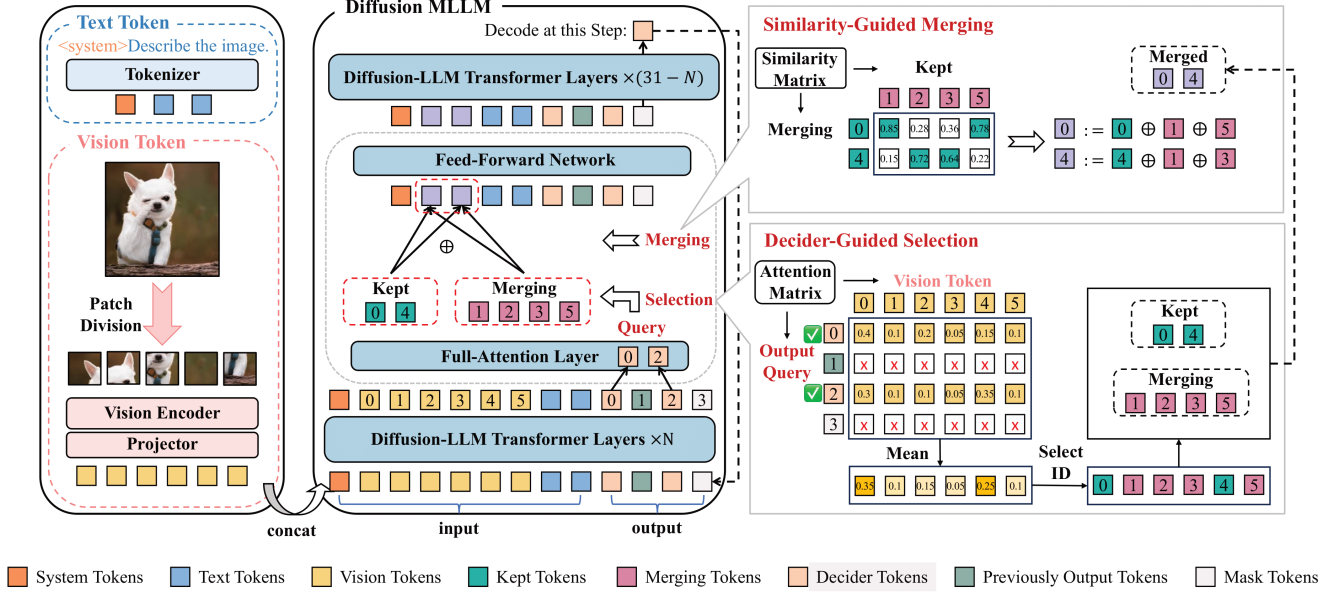


Figure 4: **The architecture of the D³ToM framework.** The main architecture (left) shows the merging operation occurring at layer l^* . The detailed process (right) illustrates the two key stages: (1) Decider-Guided Selection, where decider tokens guide the selection of visual tokens to be kept, and (2) Similarity-Guided Merging, where merging tokens are aggregated into their most similar kept tokens.

The Decider-Guided Mechanism At each diffusion step t ($1 \leq t \leq T - 1$), we designate the tokens revealed in the preceding iteration as decider tokens. Formally, we define this set as follows:

$$\mathcal{D}^{(t)} = \left\{ x_i^{(t)} \mid x_i^{(t)} \neq [\text{MASK}] \wedge x_i^{(t+1)} = [\text{MASK}] \right\}, \quad (4)$$

where $X^{(t)} = (x_1^{(t)}, \dots, x_O^{(t)})$ denotes the partially decoded sequence at step t and $x_i^{(t)}$ is its i -th token. At the designated merge layer l^* , let $\mathcal{H}^{(l^*, t)} \in \mathbb{R}^{N \times d_{\text{model}}}$ denote the token representations at step t , and let $W_Q, W_K \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ be the query and key projection matrices. We then compute the self-attention matrix:

$$A = \text{softmax} \left(\frac{(\mathcal{H}^{(l^*, t)} W_Q) (\mathcal{H}^{(l^*, t)} W_K)^\top}{\sqrt{d_{\text{model}}}} \right), \quad (5)$$

where $A_{i,j}$ denotes the attention weight from output token $x_i^{(t)}$ (query position) to visual token $v_j^{(t)}$ (key position).

We quantify the importance of each visual token $v_j^{(t)}$ by aggregating the attention it receives from all decider tokens in $\mathcal{D}^{(t)}$:

$$S_j^{(t)} = \sum_{x_i^{(t)} \in \mathcal{D}^{(t)}} A_{i,j}. \quad (6)$$

Based on these scores, we partition the set of all visual tokens $\mathcal{V}$ into two disjoint subsets: the tokens to be kept, $\mathcal{V}_{\text{kept}}(t)$, and those to be merged, $\mathcal{V}_{\text{merge}}(t)$. Let $\text{Rank}(S_j^{(t)})$

be the rank of token v_j according to its score. We define these sets as:

$$\begin{aligned} \mathcal{V}_{\text{kept}}(t) &:= \{v_j \in \mathcal{V} \mid \text{Rank}(S_j^{(t)}) \leq (1 - \alpha_t)|V|\}, \\ \mathcal{V}_{\text{merge}}(t) &:= \mathcal{V} \setminus \mathcal{V}_{\text{kept}}(t). \end{aligned} \quad (7)$$

To prepare for the next iteration, we update the decider set to exclusively include the most recently revealed tokens. This ensures that only newly decoded tokens guide the next importance computation.

Similarity-Based Merging Given a merge ratio α_t , our method merges the tokens in $\mathcal{V}_{\text{merge}}(t)$ into their nearest neighbors within $\mathcal{V}_{\text{kept}}(t)$, based on cosine similarity. This process reduces the sequence length from its pre-merge value, N , to a post-merge length, $N_m(t)$, given by:

$$N_m(t) = N - \alpha_t |V|. \quad (8)$$

The merging operates on the post-attention hidden states $\tilde{\mathcal{H}}^{(l^*, t)} \in \mathbb{R}^{N \times d_{\text{model}}}$ from layer l^* , where $\tilde{\mathcal{H}}^{(l^*, t)} = (\tilde{h}_1^{(t)}, \dots, \tilde{h}_N^{(t)})$ and each $\tilde{h}_j^{(t)} \in \mathbb{R}^{d_{\text{model}}}$ denotes the post-attention representation of token.

For each token $v_m^{(t)} \in \mathcal{V}_{\text{merge}}(t)$, we identify its most similar counterpart in the kept set using cosine similarity:

$$k^* = \arg \max_{v_k^{(t)} \in \mathcal{V}_{\text{kept}}(t)} \frac{\tilde{h}_m^{(t)} \cdot \tilde{h}_k^{(t)}}{\|\tilde{h}_m^{(t)}\|_2 \|\tilde{h}_k^{(t)}\|_2}. \quad (9)$$

The merging step then aggregates the representations:

$$\tilde{h}_{k^*}^{(t)} \leftarrow \tilde{h}_{k^*}^{(t)} + \tilde{h}_m^{(t)}. \quad (10)$$

Finally, we physically remove the merged token positions from the hidden state matrix, yielding a shortened tensor $\tilde{\mathcal{H}}_{\text{merge}}^{(t)} \in \mathbb{R}^{N_m(t) \times d_{\text{model}}}$. This tensor then propagates through the subsequent layers, reducing the computational complexity for these layers.

Timestep-Dependent Merge Schedule To further align token merging with the evolving nature of the diffusion process, we introduce a timestep-aware variant, D³ToM-t, in which the merge ratio α_t varies with the denoising timestep. Intuitively, early iterations construct a coarse semantic draft and therefore benefit from high token retention, whereas later iterations focus on detail refinement and can tolerate more aggressive merging.

Let $\alpha_{\min}$ and $\alpha_{\max}$ denote the minimum and maximum merge ratios allowed during inference. We define a linear schedule:

$$\alpha_t = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \frac{t-1}{T-1}, \quad 1 \leq t \leq T. \quad (11)$$

Under this schedule, α_t is smallest at the beginning of decoding and largest near the end, merging more redundant tokens.

4 Experiments

4.1 Experimental Setup

Models and Baselines We implement D³ToM on LaViDa (Li et al. 2025) and compare it with representative autoregressive token-reduction methods including FastV (Chen et al. 2024), PyramidDrop (Xing et al. 2024), and VisionZip (Yang et al. 2025d), adapted to the diffusion setting.

Benchmarks. We evaluate our method on 7 multimodal benchmarks including MMMU (Yue et al. 2024), MM-Bench (Liu et al. 2024b), LLaVA-Bench (Liu et al. 2024a), GQA (Hudson and Manning 2019), ScienceQA (Saikh et al. 2022), AI2D (Hiippala et al. 2021) and MathVision (Ahmad et al. 2025).

4.2 Main Results

Table 1 demonstrates that D³ToM and its timestep-aware variant D³ToM-t consistently preserve more task accuracy than all competing compression strategies under every token-retention regime. When we retain only a quarter of the visual tokens, D³ToM-t maintains over 98% of the original LaViDa performance. At the extreme 10% retained setting, it retains 96.75%.

The dynamic schedule in D³ToM-t provides consistent gains over the fixed-ratio version across all budgets, indicating that our decider-guided merging preserves critical visual information even under aggressive compression. Collectively, these results confirm that D³ToM delivers state-of-the-art performance retention for diffusion-based MLLMs across a broad spectrum of multimodal tasks.

4.3 Efficiency Analysis

Efficientness Evaluation We begin by defining the core parameters for our analysis: d is the hidden size, m is the

Algorithm 1: D³ToM Denoising with Decider-Guided Dynamic Token Merging

```

1: Input: visual tokens  $V$ , prompt tokens  $P$ , total denoising steps  $T$ , total layers  $L_{\text{model}}$ , merge layer  $l^*$ , merge schedule  $\{\alpha_t\}_{t=1}^T$ 
2: Output: final decoded sequence  $X^{(0)}$ 
3: Init:  $X^{(T)} \leftarrow ([\text{MASK}], \dots, [\text{MASK}])$ ,  $\mathcal{D}^{(T)} \leftarrow \emptyset$ 
4: Let  $\mathcal{V}$  be the index set of visual tokens in the concatenated sequence
5: for  $t = T$  down to 1 do
6:    $\mathcal{H}^{(0,t)} \leftarrow \text{Embed}(\text{Concat}(V, P, X^{(t)}))$ 
7:   for  $l = 1$  to  $L_{\text{model}}$  do
8:      $(\tilde{\mathcal{H}}^{(l,t)}, A^{(l,t)}) \leftarrow \text{SelfAttn}_l(\mathcal{H}^{(l-1,t)})$ 
9:     if  $l = l^*$  and  $\mathcal{D}^{(t)} \neq \emptyset$  then
10:        $\alpha \leftarrow \alpha_t$ 
11:        $S_j^{(t)} \leftarrow \sum_{x_i^{(t)} \in \mathcal{D}^{(t)}} A_{i,j}^{(l,t)}, \quad \forall j \in \mathcal{V}$ 
12:        $\mathcal{V}_{\text{kept}}(t) \leftarrow \text{TopK}_{(1-\alpha)|\mathcal{V}|}(S^{(t)})$ 
13:        $\mathcal{V}_{\text{merge}}(t) \leftarrow \mathcal{V} \setminus \mathcal{V}_{\text{kept}}(t)$ 
14:       for all  $v_m \in \mathcal{V}_{\text{merge}}(t)$  do
15:          $k^* \leftarrow \arg \max_{v_k \in \mathcal{V}_{\text{kept}}(t)} \cos(\tilde{h}_m^{(l,t)}, \tilde{h}_k^{(l,t)})$ 
16:          $\tilde{h}_{k^*}^{(l,t)} \leftarrow \tilde{h}_{k^*}^{(l,t)} + \tilde{h}_m^{(l,t)}$ 
17:       end for
18:       Remove positions in  $\mathcal{V}_{\text{merge}}(t)$  from  $\tilde{\mathcal{H}}^{(l,t)}$  to obtain  $\hat{\mathcal{H}}^{(l,t)}$ 
19:     else
20:        $\hat{\mathcal{H}}^{(l,t)} \leftarrow \tilde{\mathcal{H}}^{(l,t)}$ 
21:     end if
22:      $\mathcal{H}^{(l,t)} \leftarrow \text{FFN}_l(\hat{\mathcal{H}}^{(l,t)})$ 
23:   end for
24:    $\tilde{X} \leftarrow \text{Decode}(\mathcal{H}^{(L_{\text{model}},t)})$ 
25:    $X^{(t-1)} \leftarrow \mathcal{S}(\tilde{X}, X^{(t)}, t)$ 
26:    $\mathcal{D}^{(t-1)} \leftarrow \{x_i^{(t-1)} \mid x_i^{(t-1)} \neq [\text{MASK}], x_i^{(t)} = [\text{MASK}]\}$ 
27: end for
28: return  $X^{(0)}$ 

```

intermediate feed-forward width, L is the number of layers, and T is the number of denoising steps. The total token count N per step is given by:

$$N = |V| + |P| + |O|. \quad (12)$$

The computational cost for a single Transformer layer processing n tokens, denoted $\text{FLOPs}_{\text{layer}}(n)$, consists of self-attention and feed-forward network (FFN) components, as follows:

$$\text{FLOPs}_{\text{layer}}(n) = \underbrace{4nd^2 + 2n^2d}_{\text{self-attention}} + \underbrace{3ndm}_{\text{FFN}}. \quad (13)$$

Consequently, the total cost for a baseline forward pass without merging is:

$$\text{FLOPs}_{\text{baseline}} = T \times L \times \text{FLOPs}_{\text{layer}}(N). \quad (14)$$

For our method, D³ToM, we introduce a merge ratio $\alpha_t \in [0, 1]$ at each step t , yielding a reduced sequence length

Method	MMMU	SQA	MMB	MathVision	AI2D	LLaVA-B	GQA	Avg.↑
Upper Bound, Retain 100% Tokens								
LaViDa	43.78	72.34	74.24	20.39	69.46	71.60	66.20	100.0%
Retain 50% Tokens								
FastV	40.68	69.16	64.37	16.85	62.85	58.70	56.20	87.89%
PDrop	41.45	70.64	66.45	17.12	64.24	62.90	60.30	91.03%
VisionZip	42.67	70.30	68.18	17.96	66.06	63.30	59.40	92.54%
D³ToM	43.00	72.38	71.97	18.75	67.76	70.20	63.00	96.85%
D³ToM-t	42.78	72.53	72.73	20.39	66.71	70.80	63.20	98.05%
Retain 33.3% Tokens								
FastV	40.87	66.47	62.75	15.46	60.38	52.60	52.80	83.68%
PDrop	41.74	70.16	64.08	17.67	64.51	62.40	57.80	90.38%
VisionZip	42.11	70.10	68.18	18.45	65.58	62.10	57.60	91.94%
D³ToM	43.11	71.92	70.70	19.28	66.71	71.00	62.40	96.73%
D³ToM-t	42.67	72.43	73.48	19.74	65.67	71.40	63.20	97.59%
Retain 25% Tokens								
FastV	40.12	61.87	60.38	15.13	57.74	49.70	50.40	80.20%
PDrop	40.59	69.47	62.20	17.54	64.19	59.80	56.40	88.53%
VisionZip	40.78	70.90	64.40	18.39	65.38	58.60	58.20	90.28%
D³ToM	42.33	71.64	69.70	18.68	66.19	70.80	60.40	95.23%
D³ToM-t	43.00	72.48	73.48	20.07	65.45	72.10	63.60	98.12%
Retain 16.7% Tokens								
FastV	38.13	57.45	58.46	14.26	56.68	48.90	50.10	77.25%
PDrop	40.85	68.74	60.83	17.28	63.14	60.60	56.20	87.92%
VisionZip	41.56	70.75	67.42	17.73	65.25	58.90	57.20	90.44%
D³ToM	41.78	72.19	71.21	18.85	65.67	71.30	62.60	96.04%
D³ToM-t	43.11	72.73	73.48	20.07	65.67	71.90	63.40	98.16%
Retain 10% Tokens								
FastV	38.42	55.38	57.74	14.42	55.40	49.20	48.40	76.34%
PDrop	40.27	66.95	60.17	17.74	63.27	55.70	55.20	86.41%
VisionZip	40.78	70.80	65.91	18.09	65.45	54.30	54.40	88.68%
D³ToM	42.67	71.44	68.93	18.52	65.87	70.10	62.60	95.31%
D³ToM-t	42.89	72.58	73.48	18.68	65.67	70.20	63.40	96.75%

Table 1: Performance evaluation on standard multimodal benchmarks. Scores are reported for each benchmark, along with the average performance retention relative to the vanilla model (100%).

$N_m(t) = N - \alpha_t |V|$. This operation introduces a small computational overhead:

$$\text{FLOPs}_{\text{merge}}(t) = 2\alpha_t(1 - \alpha_t) |V|^2 d + \alpha_t |V| d. \quad (15)$$

The total per-step cost, with merging applied at layer l^* , is the sum of costs from the pre-merge layers ($< l^*$), the split-computation layer l^* , the post-merge layers ($> l^*$), and the merge overhead itself:

$$\begin{aligned} \text{FLOPs}_{\text{step}}(t) = & l^* \times \text{FLOPs}_{\text{layer}}(N) \\ & + [\text{FLOPs}_{\text{attn}}(N) + \text{FLOPs}_{\text{ffn}}(N_m(t))] \\ & + (L - l^* - 1) \times \text{FLOPs}_{\text{layer}}(N_m(t)) \\ & + \text{FLOPs}_{\text{merge}}(t). \end{aligned} \quad (16)$$

The total computational cost for D³ToM is the sum of these per-step costs over the entire trajectory:

$$\text{FLOPs}_{\text{D}^3\text{ToM}} = \sum_{t=1}^T \text{FLOPs}_{\text{step}}(t). \quad (17)$$

When comparing a constant merge ratio $\bar{\alpha}$ with a time-varying linear schedule $\{\alpha_t\}$ that has the same mean, the primary difference in FLOPs, Δ , arises only from the quadratic self-attention term:

$$\Delta = 4d|V|^2 [(T - 1) \text{Var}(\alpha_t)]. \quad (18)$$

This difference is negligible when $|V| \ll N$, satisfying $\Delta/(4dN^2) < 1\%$. This proves that the cost of the time-varying schedule is approximately equal to that of the constant-ratio configuration:

$$\text{FLOPs}_{\text{D}^3\text{ToM-t}} \simeq \text{FLOPs}_{\text{D}^3\text{ToM}}. \quad (19)$$

We derive the FLOPs expressions for all baseline methods under the same diffusion-MLLM assumptions. To ensure a fair comparison, we match their total effective token budget, meaning the cumulative sequence length over all denoising steps, to that of D³ToM. In this way, any observed differences reflect the algorithmic behavior rather than discrepancies in token counts.

Method	TFLOPs↓		Time (s)↓	
	Abs.	Rel.	Abs.	Rel.
Upper Bound, Retain 100% tokens				
LaViDa	262.60	100%	1156.65s	100%
Retain 50% tokens				
FastV	158.10	60.2%	786.42s	68.0%
PDrop	187.18	71.3%	845.61s	73.1%
VisionZip	143.57	54.7%	764.14s	66.1%
D³ToM	159.42	60.7%	766.40s	66.3%
D³ToM-t	160.03	60.9%	764.89s	66.1%
Retain 33.3% tokens				
FastV	124.01	47.2%	667.24s	57.7%
PDrop	137.13	52.2%	718.06s	62.1%
VisionZip	104.74	39.9%	636.99s	55.1%
D³ToM	125.73	47.9%	664.89s	57.5%
D³ToM-t	125.99	48.0%	667.30s	57.7%
Retain 25% tokens				
FastV	107.22	40.8%	612.72s	53.0%
PDrop	119.57	45.6%	661.43s	57.2%
VisionZip	85.62	32.6%	557.85s	48.2%
D³ToM	109.12	41.6%	604.87s	52.3%
D³ToM-t	109.27	41.6%	604.68s	52.3%
Retain 16.7% tokens				
FastV	90.54	34.5%	554.13s	47.9%
PDrop	105.87	40.3%	609.27s	52.7%
VisionZip	66.62	25.4%	504.37s	43.6%
D³ToM	92.61	35.3%	550.37s	47.6%
D³ToM-t	92.68	35.3%	552.14s	47.8%
Retain 10% tokens				
FastV	77.14	29.4%	487.19s	42.1%
PDrop	97.16	37.0%	578.31s	50.0%
VisionZip	51.36	19.6%	420.01s	36.3%
D³ToM	79.35	30.2%	492.70s	42.6%
D³ToM-t	79.37	30.2%	493.88s	42.7%

Table 2: Inference cost under Sec.4.3. Values are averaged over LLaVA-Bench with $T=32$ and $O=64$.

Efficientness Results We evaluate the computational efficiency of our method using both theoretical FLOPs and measured inference time. The results reported in Table 2 are obtained by running LLaVA-Bench on a single NVIDIA A6000 GPU.

Compared with inner-LLM pruning baselines, D³ToM and D³ToM-t use essentially the same FLOPs and inference time as FastV and remain below PDrop. At the 10% retention point, D³ToM-t consumes about 30% of the reference FLOPs and 42% of the wall-clock time while delivering higher accuracy. VisionZip lowers raw cost further by deleting tokens in visual encoder, but its one-shot pruning cannot follow the evolving visual saliency during diffusion process. By adjusting token selection at every denoising step, our decider-guided merging maintains competitive efficiency without the performance limitations.

Method*	MMMU	SQA	MMB	AI2D	Avg.↑
Upper Bound, Retain 100% Tokens					
LaViDa*	43.78	71.34	70.44	69.82	100%
Retain 50% Tokens					
D ³ ToM*	42.89	71.39	70.45	69.92	99.5%
D ³ ToM-t*	43.00	71.19	70.36	69.98	99.5%
Retain 33.3% Tokens					
D ³ ToM*	43.00	71.29	70.10	69.88	99.4%
D ³ ToM-t*	42.56	71.19	70.27	69.79	99.2%
Retain 25% Tokens					
D ³ ToM*	43.22	71.54	70.28	69.62	99.6%
D ³ ToM-t*	42.67	70.47	70.36	69.92	99.1%
Retain 16.7% Tokens					
D ³ ToM*	43.22	71.59	70.10	69.72	99.6%
D ³ ToM-t*	42.44	71.34	70.45	69.75	99.2%
Retain 10% Tokens					
D ³ ToM*	43.67	71.69	69.50	69.66	99.7%
D ³ ToM-t*	42.78	71.24	70.36	69.92	99.4%

Table 3: Compatibility with KV-Cache: * indicates that the method is augmented with the Prefix-DLM Cache.

4.4 Compatibility with KV-Cache

During cached decoding the model stores key and value tensors $K, V \in \mathbb{R}^{(|V|+|P|) \times d_{\text{model}}}$. After importance filtering we obtain the kept index set $\mathcal{K}$ and its complement $\mathcal{M}$. For every $m \in \mathcal{M}$ we route it to the most similar kept token

$$\pi(m) = \arg \max_{k \in \mathcal{K}} \frac{K_m \cdot K_k}{\|K_m\|_2 \|K_k\|_2}. \quad (20)$$

We then merge caches and renormalise

$$\begin{aligned} K_{\pi(m)} &\leftarrow K_{\pi(m)} + K_m, \\ V_{\pi(m)} &\leftarrow V_{\pi(m)} + V_m, \end{aligned} \quad (21)$$

and finally drop the positions in $\mathcal{M}$, so subsequent attention operates on the shorter cache of length L_{kept} . The procedure is vectorised and does not modify model parameters.

Table 3 demonstrates that integrating D³ToM with the Prefix-DLM cache consistently shortens the cached sequence while retaining almost full of model accuracy. These confirm decider-guided token merging and KV caching act on distinct parts and can therefore be combined without compromising prediction quality.

5 Conclusion

We introduce D³ToM, a simple yet effective approach that physically shortens the visual token sequence in Diffusion MLLMs, thereby reducing the computational burden of denoising inference. By dynamically focusing computation on the most salient visual tokens, D³ToM preserves generation quality throughout the diffusion process. Empirical results confirm that the proposed merging strategy accelerates inference while maintaining competitive accuracy across diverse multimodal benchmarks.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (Grant No. 62471287).

References

- Ahmad, M. A.; Ahmed, T.; Aslam, M.; Rehman, A.; Alamri, F. S.; Bahaj, S. A.; and Saba, T. 2025. MathVision: An accessible intelligent agent for visually impaired people to understand mathematical equations. *IEEE Access*, 13: 6155–6165.
- Alvar, S. R.; Singh, G.; Akbari, M.; and Zhang, Y. 2025. DivPrune: Diversity-based visual token pruning for large multimodal models. In *CVPR*, 9392–9401.
- Bolya, D.; Fu, C.-Y.; Dai, X.; Zhang, P.; Feichtenhofer, C.; and Hoffman, J. 2023. Token Merging: Your ViT but faster. In *ICLR*.
- Chen, L.; Zhao, H.; Liu, T.; Bai, S.; Lin, J.; Zhou, C.; and Chang, B. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, 19–35.
- Choi, J.; Lee, S.; Chu, J.; Choi, M.; and Kim, H. J. 2024. vid-TLDR: Training free token merging for light-weight video transformer. In *CVPR*, 18771–18781.
- Feng, Z.; Xu, J.; Ma, L.; and Zhang, S. 2024. Efficient video transformers via spatial-temporal token merging for action recognition. *ACM Trans. Multim. Comput. Commun. Appl.*, 20(4): 120:1–120:21.
- Gong, S.; Agarwal, S.; Zhang, Y.; Ye, J.; Zheng, L.; Li, M.; An, C.; Zhao, P.; Bi, W.; Han, J.; Peng, H.; and Kong, L. 2025a. Scaling diffusion language models via adaptation from autoregressive models. In *ICLR*.
- Gong, S.; Zhang, R.; Zheng, H.; Gu, J.; Jaitly, N.; Kong, L.; and Zhang, Y. 2025b. DiffuCoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*.
- Hiippala, T.; Alikhani, M.; Haverinen, J.; Kalliokoski, T.; Logacheva, E.; Orekhova, S.; Tuomainen, A.; Stone, M.; and Bateman, J. A. 2021. AI2D-RST: A multimodal corpus of 1000 primary school science diagrams. *Lang. Resour. Evaluation*, 55(3): 661–688.
- Huang, W.; Zhai, Z.; Shen, Y.; Cao, S.; Zhao, F.; Xu, X.; Ye, Z.; and Lin, S. 2025a. Dynamic-LLaVA: Efficient multimodal large language models via dynamic vision-language context sparsification. In *ICLR*.
- Huang, Z.; Chen, Z.; Wang, Z.; Li, T.; and Qi, G.-J. 2025b. Reinforcing the diffusion chain of lateral thought with diffusion language models. *arXiv preprint arXiv:2505.10446*.
- Hudson, D. A.; and Manning, C. D. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 6700–6709.
- Hyun, J.; Hwang, S.; Han, S. H.; Kim, T.; Lee, I.; Wee, D.; Lee, J.-Y.; Kim, S. J.; and Shim, M. 2025. Multi-granular spatio-temporal token merging for training-free acceleration of video LLMs. *arXiv preprint arXiv:2507.07990*.
- Jeddi, A.; Baghbanzadeh, N.; Dolatabadi, E.; and Taati, B. 2025. Similarity-aware token pruning: Your VLM but faster. *arXiv preprint arXiv:2503.11549*.
- Jiang, Y.; Wu, Q.; Lin, W.; Yu, W.; and Zhou, Y. 2025. What kind of visual tokens do we need? Training-free visual token pruning for multi-modal large language models from the perspective of graph. In *AAAI*, 4075–4083.
- Kallini, J.; Murty, S.; Manning, C. D.; Potts, C.; and Csordás, R. 2025. MrT5: Dynamic token merging for efficient byte-level language models. In *ICLR*.
- Li, S.; Kallidromitis, K.; Bansal, H.; Gokul, A.; Kato, Y.; Kozuka, K.; Kuen, J.; Lin, Z.; Chang, K.-W.; and Grover, A. 2025. Lavidia: A large diffusion language model for multimodal understanding. *arXiv preprint arXiv:2505.16839*.
- Liu, H.; Li, C.; Li, Y.; and Lee, Y. J. 2024a. Improved base-lines with visual instruction tuning. In *CVPR*, 26286–26296.
- Liu, Y.; Duan, H.; Zhang, Y.; Li, B.; Zhang, S.; Zhao, W.; Yuan, Y.; Wang, J.; He, C.; Liu, Z.; Chen, K.; and Lin, D. 2024b. MMBench: Is your multi-modal model an all-around player? In *ECCV*, 216–233.
- Liu, Y.; Wang, Y.; Shi, B.; Zhang, X.; Dai, W.; Li, C.; Xiong, H.; and Tian, Q. 2025a. METEOR: Multi-encoder collaborative token pruning for efficient vision language models. *arXiv preprint arXiv:2507.20842*.
- Liu, Z.; Yang, Y.; Zhang, Y.; Chen, J.; Zou, C.; Wei, Q.; Wang, S.; and Zhang, L. 2025b. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.
- Ma, X.; Yu, R.; Fang, G.; and Wang, X. 2025. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*.
- Nie, S.; Zhu, F.; You, Z.; Zhang, X.; Ou, J.; Hu, J.; Zhou, J.; Lin, Y.; Wen, J.-R.; and Li, C. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.
- Saikh, T.; Ghosal, T.; Mittal, A.; Ekbal, A.; and Bhattacharyya, P. 2022. ScienceQA: A novel resource for question answering on scholarly articles. *Int. J. Digit. Libr.*, 23(3): 289–301.
- Shang, Y.; Cai, M.; Xu, B.; Lee, Y. J.; and Yan, Y. 2024. LLaVA-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*.
- Shao, K.; Tao, K.; Qin, C.; You, H.; Sui, Y.; and Wang, H. 2025. HoliTom: Holistic token merging for fast video large language models. *arXiv preprint arXiv:2505.21334*.
- Tschannen, M.; Gritsenko, A.; Wang, X.; Naeem, M. F.; Alabdulmohsin, I.; Parthasarathy, N.; Evans, T.; Beyer, L.; Xia, Y.; Mustafa, B.; et al. 2025. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*.
- Wei, Q.; Zhang, Y.; Liu, Z.; Liu, D.; and Zhang, L. 2025. Accelerating diffusion large language models with SlowFast: The three golden principles. *arXiv preprint arXiv:2506.10848*.
- Wen, Z.; Qu, J.; Liu, D.; Liu, Z.; Wu, R.; Yang, Y.; Jin, X.; Xu, H.; Liu, X.; Li, W.; et al. 2025. The devil behind the

mask: An emergent safety vulnerability of diffusion LLMs. *arXiv preprint arXiv:2507.11097*.

Wu, C.; Zhang, H.; Xue, S.; Liu, Z.; Diao, S.; Zhu, L.; Luo, P.; Han, S.; and Xie, E. 2025. Fast-dllm: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *arXiv preprint arXiv:2505.22618*.

Xing, L.; Huang, Q.; Dong, X.; Lu, J.; Zhang, P.; Zang, Y.; Cao, Y.; He, C.; Wang, J.; Wu, F.; et al. 2024. Pyramidrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*.

Xu, R.; Wang, Y.; Luo, Y.; and Du, B. 2025a. Rethinking visual token reduction in LVLs under cross-modal misalignment. *arXiv preprint arXiv:2506.22283*.

Xu, R.; Wang, Y.; Luo, Y.; and Du, B. 2025b. Rethinking visual token reduction in LVLs under cross-modal misalignment. *arXiv preprint arXiv:2506.22283*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, C.; Sui, Y.; Xiao, J.; Huang, L.; Gong, Y.; Li, C.; Yan, J.; Bai, Y.; Sadayappan, P.; Hu, X.; and Yuan, B. 2025b. TopV: Compatible token pruning with inference time optimization for fast and low-memory multimodal vision language model. In *CVPR*, 19803–19813.

Yang, L.; Tian, Y.; Li, B.; Zhang, X.; Shen, K.; Tong, Y.; and Wang, M. 2025c. MMaDA: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*.

Yang, S.; Chen, Y.; Tian, Z.; Wang, C.; Li, J.; Yu, B.; and Jia, J. 2025d. VisionZip: Longer is better but not necessary in vision language models. In *CVPR*, 19792–19802.

Ye, J.; Gao, J.; Gong, S.; Zheng, L.; Jiang, X.; Li, Z.; and Kong, L. 2024. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*.

Ye, J.; Xie, Z.; Zheng, L.; Gao, J.; Wu, Z.; Jiang, X.; Li, Z.; and Kong, L. 2025a. Dream 7B. <https://hkunlp.github.io/blog/2025/dream>.

Ye, W.; Wu, Q.; Lin, W.; and Zhou, Y. 2025b. Fit and prune: Fast and training-free visual token pruning for multi-modal large language models. In *AAAI*, 22128–22136.

You, Z.; Nie, S.; Zhang, X.; Hu, J.; Zhou, J.; Lu, Z.; Wen, J.-R.; and Li, C. 2025. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*.

Yu, R.; Ma, X.; and Wang, X. 2025. DIMPLE: Discrete diffusion multimodal large language model with parallel decoding. *arXiv preprint arXiv:2505.16990*.

Yue, X.; Ni, Y.; Zheng, T.; Zhang, K.; Liu, R.; Zhang, G.; Stevens, S.; Jiang, D.; Ren, W.; Sun, Y.; Wei, C.; Yu, B.; Yuan, R.; Sun, R.; Yin, M.; Zheng, B.; Yang, Z.; Liu, Y.; Huang, W.; Sun, H.; Su, Y.; and Chen, W. 2024. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, 9556–9567.

Zhang, L.; Hu, A.; Xu, H.; Yan, M.; Xu, Y.; Jin, Q.; Zhang, J.; and Huang, F. 2024a. TinyChart: Efficient chart understanding with program-of-thoughts learning and visual token merging. 1882–1898.

Zhang, Y.; Fan, C.-K.; Ma, J.; Zheng, W.; Huang, T.; Cheng, K.; Gudovskiy, D.; Okuno, T.; Nakata, Y.; Keutzer, K.; et al. 2024b. SparseVLM: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*.

Zhu, F.; Wang, R.; Nie, S.; Zhang, X.; Wu, C.; Hu, J.; Zhou, J.; Chen, J.; Lin, Y.; Wen, J.-R.; et al. 2025. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*.

Supplementary for D³ToM:

Decider-Guided Dynamic Token Merging for Accelerating Diffusion MLLMs

Shuochen Chang, Xiaofeng Zhang*, Qingyang Liu, Li Niu†

MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{csc1332741686, framebreak, narumimaria, ustcnewly}@sjtu.edu.cn

A Detailed Experimental Settings

A.1 Implementation Details

Our method is implemented on the LaViDa-LLaDA-8B models. Evaluation and inference is performed on an NVIDIA RTX A6000(48G) GPU. To ensure a fair comparison of computational cost with baseline methods, we used total denoising FLOPs and total evaluation time. Baselines are configured for comparable FLOPs: FastV prunes same proportion of tokens at same layer; PDrop keeps $1.5\times$ tokens as we kept after merge at layer 7 then pruning as same proportion with layer 7 at layers 15 and 23, respectively; VisionZip maintain a consistent proportion of input tokens with our method. To compare Inference time, All methods are evaluated on the full LLaVA-Bench set under identical decoding parameters. We report generation length N_{output} used to produce results in the main paper in Table4.

Dataset	$ O $	Dataset	$ O $
MMMU	16	AI2D	16
SQA	16	LLaVA-Bench	64
MMBench	16	GQA	16
MathVision	100		

Table 4: Settings to produce results of Table 1 at Sec4.2 in main paper. By default, we use the denoising timestep T corresponding to the $|O|$.

A.2 Datasets

We list all evaluation on these multimodal (vision-language) understanding benchmarks. We use official implementation of lmms-eval to achieve results,

Comprehensive reasoning

MMMU. The MMMU (Yue et al. 2024) benchmark is designed to access a model’s proficiency in tasks requiring expert-level and domain-specific understanding and reasoning. It consists of over 10k questions derived from college-level examinations and tutorials across science, engineering and humanities. The questions involve 30 distinct image types like diagrams and chemical structures, challenge models on both advanced perception and deep knowledge. We evaluate all methods on the validation splits.

*Project leader.

†Corresponding author.

MMBench. MMBench (Liu et al. 2024b) implements a multi-level framework to evaluate cross-specific task dimensions. The assessment is structured into 3 different levels. The hierarchical design enables a comprehensive precise analysis of the strengths and weakness across skills. We evaluate methods’ performance on dev splits.

LLaVA-Bench. LLaVA-Bench (Liu et al. 2024a) is a curated benchmark designed to access a model’s instruction-following fidelity and robustness to diverse prompt styles. The benchmark consists various categories, including conversation, detailed description, and complex reasoning. The benchmark rigorously tests a model’s weaknesses in open-ended visual question answering and detailed image interpretation. We evaluate our method’s performance on the LLaVA-Bench full benchmark.

Visual Question Answering

AI2D. The AI2D (Hiippala et al. 2021) benchmark targets the complex challenge of diagrammatic reasoning and question answering. It provides 5k diagrams sourced from textbooks, which are rich in structural elements like text labels, arrows, and pointers. The benchmark provides over 15k multiple-choice questions require models to parse this structure, integrate textual and visual information, and perform non-trivial reasoning about the diagram’s components and their relationships. We evaluate our method’s performance on the AI2D test split.

SQA. ScienceQA (Saikh et al. 2022) targets complex and multi-step reasoning with scientific domains. The benchmark features a dataset of questions across nature, language and social science. ScienceQA contains results in 26 topics, 127 categories, and 379 skills. The benchmark is explicitly designed to evaluate multimodal understanding, sophisticated reasoning chains and interpretability of model-generated answers. We evaluate methods on the image split to achieve visual ability.

GQA. The GQA (Hudson and Manning 2019) is composed of three parts: scene graphs, questions, and images. GQA offers a targeted evaluation methodology for visual question answering, grounded in detailed scene graph representations of images. The questions contained in GQA are generated to test model’s ability to reason about object attributes, the understanding of visual scenes and image spatial relationships. We evaluate all methods’ performance on GQA_lite split.

Visual Question Answering

MathVision MathVision (Ahmad et al. 2025) benchmark is a carefully curated of 3,040 image-grounded mathematical problems collected from real mathematics competitions. It spanning 16 distinct mathematical disciplines—including algebra, analytic geometry, combinatorics and probability—and organized into five clearly defined difficulty tiers. MathVision offers fine-grained test bed for evaluating MLLMs by markedly widening the topical coverage and problem diversity beyond earlier resources such as MathVista. We conduct our evaluation in Sec.4.2 on the test split.

B Detailed Theoretical FLOPs Analysis

This appendix provides the detailed theoretical FLOPs derivations for the baseline methods, as referenced in the main text. The analysis herein uses the same core parameters and assumptions established in our efficiency evaluation.

B.1 Shared Notation

For our theoretical analysis, we first establish the core parameters of the model and its inputs. We consider a Transformer architecture characterized by a hidden dimension of size d , an intermediate feed-forward network (FFN) dimension of size m , and a depth of L layers. The generative process operates over T distinct denoising steps.

At each step, the model processes a sequence comprising three components: a set of visual tokens, a set of prompt tokens, and the tokens being generated. We denote their respective counts as $|V|$, $|P|$, and $|O|$. The total sequence length, N , processed by the model in a single step is therefore the sum of these components:

$$N = |V| + |P| + |O|.$$

The computational cost for a single Transformer layer that processes a sequence of length n , which we denote by $\text{FLOPs}_{\text{layer}}(n)$, is determined by its self-attention and FFN modules. The cost is formulated as follows:

$$\text{FLOPs}_{\text{layer}}(n) = \underbrace{4nd^2 + 2n^2d}_{\text{self-attention}} + \underbrace{3ndm}_{\text{FFN}}. \quad (22)$$

The token reduction methods described are controlled by hyperparameters that determine their aggressiveness. For a standardized comparison, we define a primary controlling parameter, the **average visual-token reduction ratio**, denoted by $\alpha \in [0, 1]$, where a higher value signifies a more aggressive reduction.

Scope of FLOPs Reporting. We report all Floating Point Operations (FLOPs) in this paper as theoretical calculations. This approach allows us to precisely evaluate the algorithmic efficiency of each method, independent of hardware-specific optimizations or system-level overheads like I/O.

Our calculations are based on a fixed LaViDa-LLaDA-8B model configuration. We hold the following architectural and sequence parameters constant throughout the analysis:

$$d = 4096, \quad m = 12288, \quad L = 32, \quad T = 32,$$

$$|V| = 1000, \quad |P| = 64, \quad |O| = 64.$$

These values result in a total input sequence length of $N = |V| + |P| + |O| = 1128$ tokens for each diffusion step. In our experiments, we then analyze the impact of varying two key hyperparameters: the average visual-token reduction ratio, α , and the merge layer index, l^* .

B.2 Vanilla

In the baseline configuration, no token reduction is applied. Consequently, the model processes a constant sequence length N in every layer and at each denoising step. The total theoretical cost is therefore:

$$\text{FLOPs}_{\text{baseline}} = T \times L \times \text{FLOPs}_{\text{layer}}(N). \quad (23)$$

B.3 FastV

The FastV (Chen et al. 2024) method prunes a fixed fraction R of visual tokens at a specific layer K (where $0 \leq K < L$). This operation occurs once per denoising step. Layers with an index less than K process the full sequence of length N , while layers with an index of K or greater operate on the reduced sequence. Let $N_p = N - R|V|$ denote the sequence length after pruning. The ranking step, which determines which tokens to discard, has a cost of $2d|V|$ FLOPs per denoising step. The total cost for FastV, accumulated over T denoising steps, is composed of the computation for the initial K layers, the subsequent $L - K$ layers on the pruned sequence, and the token ranking overhead:

$$\begin{aligned} \text{FLOPs}_{\text{FastV}} = T \times & \left[K \times \text{FLOPs}_{\text{layer}}(N) \right. \\ & + (L - K) \times \text{FLOPs}_{\text{layer}}(N_p) \\ & \left. + 2d|V| \right]. \quad (24) \end{aligned}$$

B.4 PyramidDrop

PyramidDrop (Xing et al. 2024) divides the decoder into four consecutive stages of equal size. For a model with $L = 32$ layers (indexed 0–31), the stages contain $L_0 = L_1 = L_2 = L_3 = 8$ layers each. Let $V_0 = |V|$ be the initial number of visual tokens. Given a user-specified average reduction ratio α , the number of visual tokens retained after the first stage is $V_1 = 1.5(1 - \alpha)|V|$. Subsequent reduction steps use a constant retention factor $\beta = V_1/|V| = 1.5(1 - \alpha)$, resulting in $V_2 = \beta V_1$ and $V_3 = \beta V_2$ for the remaining stages. The sequence length within stage $i \in \{0, 1, 2, 3\}$ is therefore $n_i = |P| + |O| + V_i$. A ranking step, with a cost proportional to the current number of visual tokens, is performed at the transition to stages 1, 2, and 3. The total cost is the sum of computations within each of the four stages plus the overhead of the three ranking steps, accumulated over T denoising steps:

$$\begin{aligned} \text{FLOPs}_{\text{PDrop}} = T \times & \left[\sum_{i=0}^3 L_i \times \text{FLOPs}_{\text{layer}}(n_i) \right. \\ & \left. + 2d(V_0 + V_1 + V_2) \right]. \quad (25) \end{aligned}$$

B.5 VisionZip

VisionZip (Yang et al. 2025d) performs a single, one-time pruning step, discarding a fraction R of visual tokens *before* they are processed by any of the Transformer layers. The resulting sequence length, $N_z = |P| + |O| + (1 - R)|V|$, remains constant throughout all L layers and T denoising steps. No subsequent ranking or merging operations occur. The total theoretical cost is therefore:

$$\text{FLOPs}_{\text{VZip}} = T \times L \times \text{FLOPs}_{\text{layer}}(N_z). \quad (26)$$

B.6 Comparison at an Equal Average Reduction Ratio

As established in the main text, $\text{D}^3\text{ToM-t}$ use a time-varying schedule $\{\alpha_t\}$ is approximately equal to D^3ToM when using a constant average ratio. Therefore, to ensure a fair and clear comparison, we normalize all baselines against a single, constant average reduction ratio, α , which corresponds to the mean of our method’s schedule. Specifically, for **FastV**, we set its pruning layer K to match D^3ToM ’s merge layer, l^* , and its reduction fraction is set to $R = \alpha$. For **Pyramid-Drop**, its own internal average reduction ratio is set to be equal to α . For **VisionZip**, the one-shot reduction fraction is also set to $R = \alpha$.

By coupling all methods to the single parameter α , we ensure that each baseline processes, on average, the same number of visual tokens per layer. Thus, any observed differences in the total FLOPs can be attributed directly to the intrinsic costs of each method’s reduction mechanism (e.g., ranking overhead) rather than to discrepancies in their overall token processing budgets.

C Ablation Studies

C.1 Impact of Merge Layer l^* and Merge Ratio α on FLOPs

This section analyzes how the merge layer index, l^* , and the merge ratio, α , jointly affect the total computational cost in FLOPs.

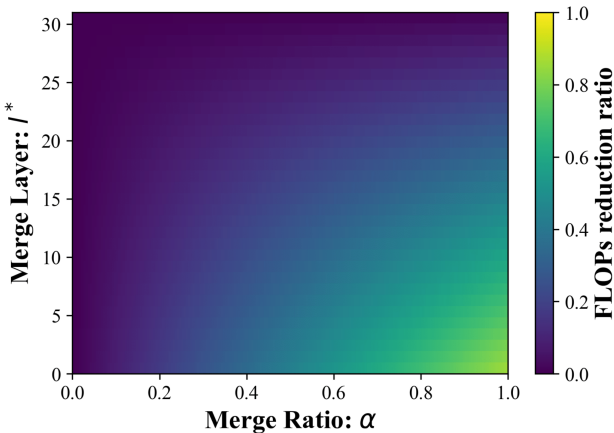


Figure 5: FLOPs reduction (%) as a function of merge layer l^* and merge ratio α . Higher is better.

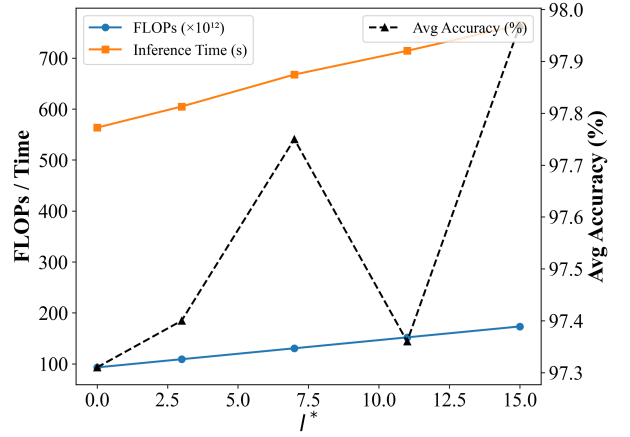


Figure 6: Ablation study on the merge layer l^* for fixed merge ratio $\alpha = 0.75$. The solid lines show theoretical FLOPs ($\times 10^{12}$) and single-GPU inference time (s) as l^* varies, and the dashed line shows average accuracy (%) across four benchmarks.

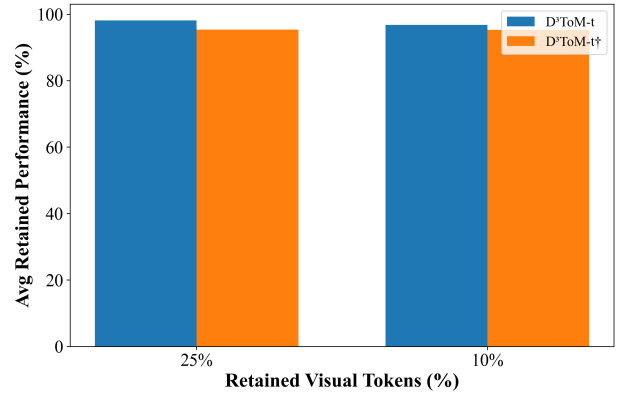


Figure 7: Comparison of average retained performance (%) for $\text{D}^3\text{ToM-t}$ (low $\rightarrow$ high) versus $\text{D}^3\text{ToM-t}^\dagger$ (high $\rightarrow$ low) merge schedules under 25% and 10% token retention.

Figure 5 plots the theoretical FLOPs reduction as a function of the merge layer, l^* , and the merge ratio, α . The reduction in computational cost increases monotonically with a higher merge ratio and an earlier merge layer. However, the gains are not uniform across the parameter space. The most significant efficiency improvements are realized when merging occurs within the initial layers (e.g., $l^* < 10$), as deferring the merge to deeper layers yields progressively smaller marginal benefits. This suggests an early, but not necessarily immediate, merge layer may provide a strong balance between efficiency and model stability.

Concurrently, the analysis of the merge ratio, α , shows that values below approximately 0.5 result in only modest computational savings. The most substantial efficiency gains begin to accumulate only as α surpasses this threshold. This

Method	FLOPs↓	Time(s)↓	Benchmark Accuracy (%)				Avg↑
			MMMU↑	LLaVA-Bench↑	SQA↑	AI2D↑	
Upper Bound, Retain 100% Tokens							
LaViDa	262.60 (100%)	1156.65s (100%)	43.78	71.60	72.34	69.46	100%
Retain 25% Tokens ($\alpha = 0.75$)							
$l^* = 0$	93.05 (35.43%)	563.49s (48.72%)	42.22	70.70	71.58	66.05	97.31%
$l^* = 3$	109.12 (41.55%)	604.87s (52.30%)	42.33	70.80	71.64	66.19	97.40%
$l^* = 7$	130.55 (49.71%)	667.56s (57.71%)	42.11	71.20	72.24	66.34	97.75%
$l^* = 11$	151.98 (57.88%)	714.16s (61.74%)	42.11	70.20	71.84	66.61	97.36%
$l^* = 15$	173.41 (66.04%)	764.76s (66.12%)	42.67	71.10	72.04	66.71	97.97%
Retain 10% Tokens ($\alpha = 0.90$)							
$l^* = 0$	60.16 (22.91%)	523.90s (45.29%)	42.00	70.20	70.78	65.74	96.62%
$l^* = 3$	79.35 (30.32%)	550.37s (47.58%)	42.67	70.10	71.44	65.87	97.19%
$l^* = 7$	104.93 (39.96%)	601.18s (51.98%)	42.44	70.30	71.67	66.27	97.40%
$l^* = 11$	130.51 (49.70%)	655.83s (56.70%)	41.78	70.30	71.54	66.45	97.04%
$l^* = 15$	156.09 (59.44%)	701.24s (60.63%)	43.22	71.10	71.68	66.48	98.21%

Table 5: Ablation study on the merge layer index l^* for fixed merge ratios $\alpha = 0.75$ and $\alpha = 0.90$. Each row lists the theoretical FLOPs ($\times 10^{12}$), single-GPU inference time (s), and accuracy (%) on four benchmarks. Inference time is measured on the LLaVA-Bench benchmark with settings $T = 32$ and $N_{output} = 64$.

indicates that the most impactful and practical range for token merging starts where the trade-off between the number of tokens removed and the computational resources saved becomes most favorable.

C.2 Ablation Study on Merge Layer l^*

Table 5 benchmarks D³ToM under a constant retention ratio of $\alpha = 0.75$ (i.e. 25% of visual tokens retained) while sweeping l^* . We report average accuracy (%) on four representative datasets and theoretical FLOPs ($\times 10^{12}$) with end-to-end wall-clock time (seconds) on LLaVA-Bench.

We select $l^* = 3$ as the default merge layer as it represents the optimal Pareto point between computational cost and model accuracy. Merging at earlier layers (e.g., $l^* = 0$) causes a notable degradation in performance, while deferring the merge to later layers ($l^* \geq 7$) provides only marginal accuracy improvements at the cost of substantial efficiency losses. Figure 6 shows that theoretical FLOPs and inference time both increase with l^* . We choose $l^* = 3$ and exhibits results in main paper.

C.3 Ablation Study on Timestep-Dependent Merge Schedule

This ablation isolates the influence of the merge-ratio direction. D³ToM-t applies a low $\rightarrow$ high linear schedule so that early diffusion steps retain most visual tokens while later steps merge aggressively. D³ToM-t[†] reverses the schedule, beginning with the strongest compression and relaxing it over time. Both variants are calibrated such that the mean retention equals the constant 25% or 10% settings, leaving the cumulative computational budget unchanged. Results are reported on the seven evaluation benchmarks used in the main paper; the last column gives the average performance retained relative to the full-token LaViDa baseline.

Result in Table 6 demonstrate that D³ToM-t outperform the high $\rightarrow$ low linear schedule. Figure 7 shows that D³ToM-t consistently yields higher average retained performance than D³ToM-t[†] under both 25% and 10% retention settings.

D Compatibility with Flash Attention

Our D³ToM method is designed for full compatibility with memory-efficient attention mechanisms such as FlashAttention. The primary challenge is to obtain the decider-guided importance scores for every visual token without materializing the full $N \times N$ attention matrix. Recall that the importance score for a visual token j , denoted $S_j^{(t)}$, is the sum of attention it receives from all decider tokens in the set $\mathcal{D}^{(t)}$, defined as:

$$S_j^{(t)} = \sum_{d_i \in \mathcal{D}^{(t)}} A_{i,j}. \quad (27)$$

To compute these scores efficiently, we propose a **dual-pass extraction** scheme. This method involves two sequential FlashAttention calls on the merge layer l^* , both sharing the same key matrix K .

D.1 Pass 1: Standard Attention Computation

The first pass executes a standard FlashAttention operation using the original query (Q), key (K), and value (V) matrices. This computes the standard output hidden states, $\mathcal{H}'$, which we cache for the subsequent merge step. The operation is as follows:

$$\mathcal{H}' = \text{FlashAttention}(Q, K, V). \quad (28)$$

D.2 Pass 2: Decider-Guided Score Extraction

The second pass is designed specifically to extract the importance scores. We first construct a new query matrix, $Q^\dagger$,

Method	MMMU	SQA	MMB	MathVision	AI2D	LLaVA-B	GQA	Avg.↑
Upper Bound, Retain 100% Tokens								
LaViDa	43.78	72.34	74.24	20.39	69.46	71.60	66.20	100.00%
Retain 25% Tokens								
D ³ ToM-t	43.00	72.48	73.48	20.07	65.45	72.10	63.60	98.12%
D ³ ToM-t [†]	42.89	72.53	71.97	18.65	66.06	65.40	62.40	95.33%
Retain 10% Tokens								
D ³ ToM-t	42.89	72.58	73.48	18.68	65.67	70.20	63.40	96.75%
D ³ ToM-t [†]	42.78	72.53	71.21	18.79	65.48	66.90	61.80	95.30%

Table 6: Impact of the linear merge-schedule direction. D³ToM-t adopts a low $\rightarrow$ high retention curve, whereas D³ToM-t[†] reverses it. All values are placeholders to be filled with measured accuracies; FLOPs are identical by construction and are therefore omitted.

composed exclusively of the hidden states of the decoder tokens:

$$Q^\dagger = Q[\mathcal{D}^{(t)}, :] \in \mathbb{R}^{|\mathcal{D}^{(t)}| \times d_{\text{model}}}. \quad (29)$$

Next, we create a modified value matrix, $V^\dagger$, where every row is set to the first basis vector $e_1 \in \mathbb{R}^{d_{\text{model}}}$:

$$\text{row}_j(V^\dagger) = e_1, \quad j = 1, \dots, N. \quad (30)$$

We then execute FlashAttention with the modified query and value matrices while reusing the original key matrix K . This yields an output tensor $O^\dagger$:

$$O^\dagger = \text{FlashAttention}(Q^\dagger, K, V^\dagger) \in \mathbb{R}^{|\mathcal{D}^{(t)}| \times d_{\text{model}}}. \quad (31)$$

Because every row in $V^\dagger$ is identical, the first channel of the output $O^\dagger$ effectively encodes the attention distribution from each decoder token to all other tokens. By aggregating this first channel across all decoder outputs, we recover the exact importance score $S_j^{(t)}$ for each visual token j as follows:

$$S_j^{(t)} = \sum_{d_i \in \mathcal{D}^{(t)}} A_{i,j} = \sum_{i=1}^{|\mathcal{D}^{(t)}|} (O_i^\dagger)_{1,j}. \quad (32)$$

D.3 Token Selection and Merging

With the importance scores $\{S_j^{(t)}\}$ computed, we perform a top-k selection to partition the visual tokens into a kept set, $\mathcal{V}_{\text{kept}}(t)$, and a merge set, $\mathcal{V}_{\text{merge}}(t)$. The physical merging operation is then applied to the cached hidden states $\mathcal{H}'$ obtained from Pass 1.

The entire dual-pass scheme introduces a negligible overhead. Since the merge occurs after the self-attention step of layer l^* , all subsequent layers operate on a physically shorter sequence. The total added cost is limited to one extra FlashAttention call and a linear-time merge operation, which is insignificant relative to the overall model FLOPs.

E Generalization Analysis

E.1 Visual Encoder Generalization

Our D³ToM method is agnostic to the choice of the visual encoder. This independence stems from its core design: the

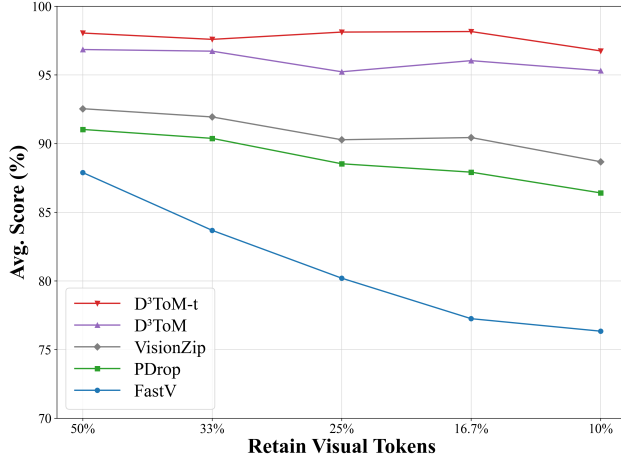
token merging module operates exclusively on the sequence of visual embeddings after they have been projected into the shared language embedding space. Consequently, no gradients or control signals are propagated back to the vision tower. This ensures that architectural differences in the encoder, such as patching strategies, positional encodings, or pre-training objectives (e.g., CLIP versus SigLIP), do not affect the merge logic.

This design allows various encoders to be used interchangeably. For example, the models evaluated in our work, LLaDA-V and LaViDa, use a SigLIP-400M encoder. However, other common encoders, such as the CLIP-ViT series frequently adopted by LLaVA-style models, are also directly compatible. The practical benefit is that no hyperparameter re-tuning or architectural search for our method is necessary when swapping visual front-ends; only the absolute number of visual tokens changes, not the downstream interface. Our empirical results confirm that after normalizing for token counts, the merge schedules remain consistent across different encoders. This suggests that future work could extend our method to other architectures, such as large-stride ConvNext or Swin models, by simply adjusting the projector’s dimensions while keeping the core merge hyperparameters of D³ToM unchanged.

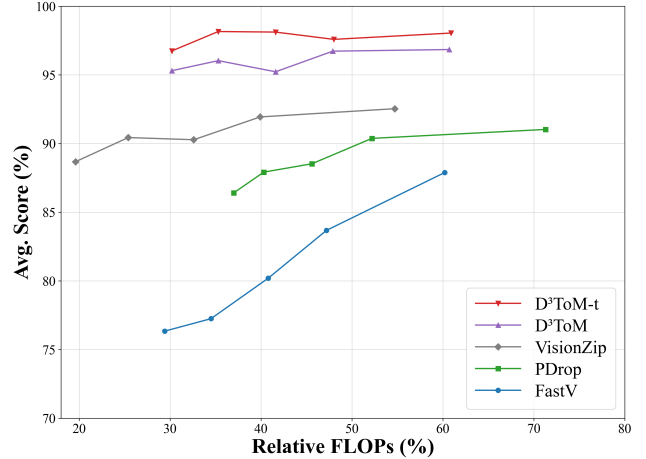
E.2 Diffusion-based MLLM Generalization

D³ToM also generalizes seamlessly across different Diffusion-based Multimodal Language Models (MLLMs), provided they are built upon the same language model backbone. We demonstrate this by applying our method to both LaViDa-LLaDA and LLaDA-V. Although these models differ in their visual front-ends and training curricula, they share the same LLaDA-8B diffusion language model. This results in an identical internal attention structure and token embedding space, which is the environment where our module operates.

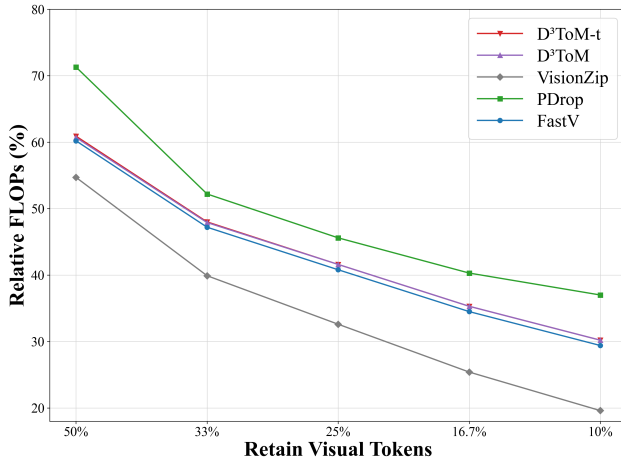
The architectural consistency between these models is documented in their respective publications. For example, LLaDA-V explicitly “maps visual features into the LLaDA language embedding space”, and LaViDa confirms that “we explored LLaDA-8B as our diffusion language model”. Be-



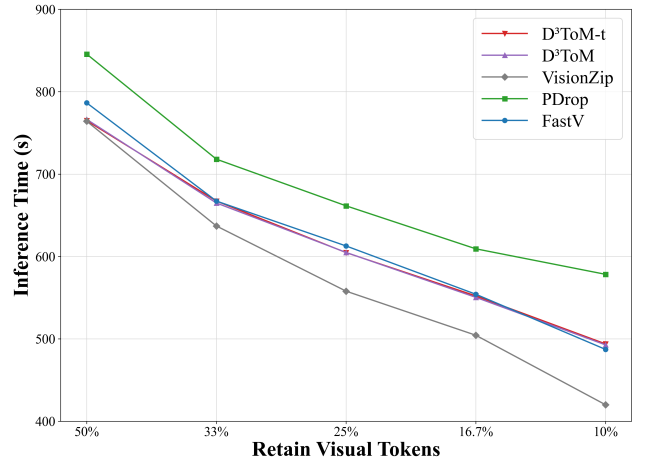
(a) Accuracy versus retained tokens



(b) Accuracy versus relative FLOPs



(c) Relative FLOPs versus retained tokens



(d) Inference latency versus retained tokens

Figure 8: Efficiency plots derived directly from the main-paper tables. All curves share identical marker, colour, and font conventions for consistency with the quantitative results.

cause our merge module operates directly within this shared Transformer stack, porting D³ToM from one model to the other requires zero code modifications or changes to its hyperparameters, including the merge layer index l^* and the merge schedule α_t . As a result, the speed-quality trade-offs established on LaViDa-LLaDA are directly transferable to LLaDA-V. This robustness could be further validated in future work by applying D³ToM with the same hyperparameters to other supported backbones, such as Dream-7B, to confirm its generalization across different model architecture.

F Extended Efficiency Analysis

This section visualises the cost–accuracy trade-off of the five compression baselines—FastV, PDrop, VisionZip,

D³ToM, and D³ToM-t. LaViDa is omitted to avoid axis stretching and to focus the discussion on the practical operating regime (10%–50% token budgets).

Figure 8(a) confirms that accuracy decreases monotonically as fewer visual tokens are kept. The slope, however, varies markedly among methods. FastV drops by 11.6% when the budget contracts from 50% to 10%, whereas D³ToM-t degrades by only 1.3%, retaining 96.75% of the full-model score at the most aggressive 10% setting.

Figure 8(b) casts the same results against relative compute. D³ToM-t achieves 96.75% accuracy at 30.2% FLOPs, while VisionZip reaches merely 88.68% at a similar 19.6–32.6% compute envelope and PDrop stays below 91.03% even at 52.2% FLOPs. The convex hull of the plot therefore places D³ToM-t on the Pareto frontier across the examined range.

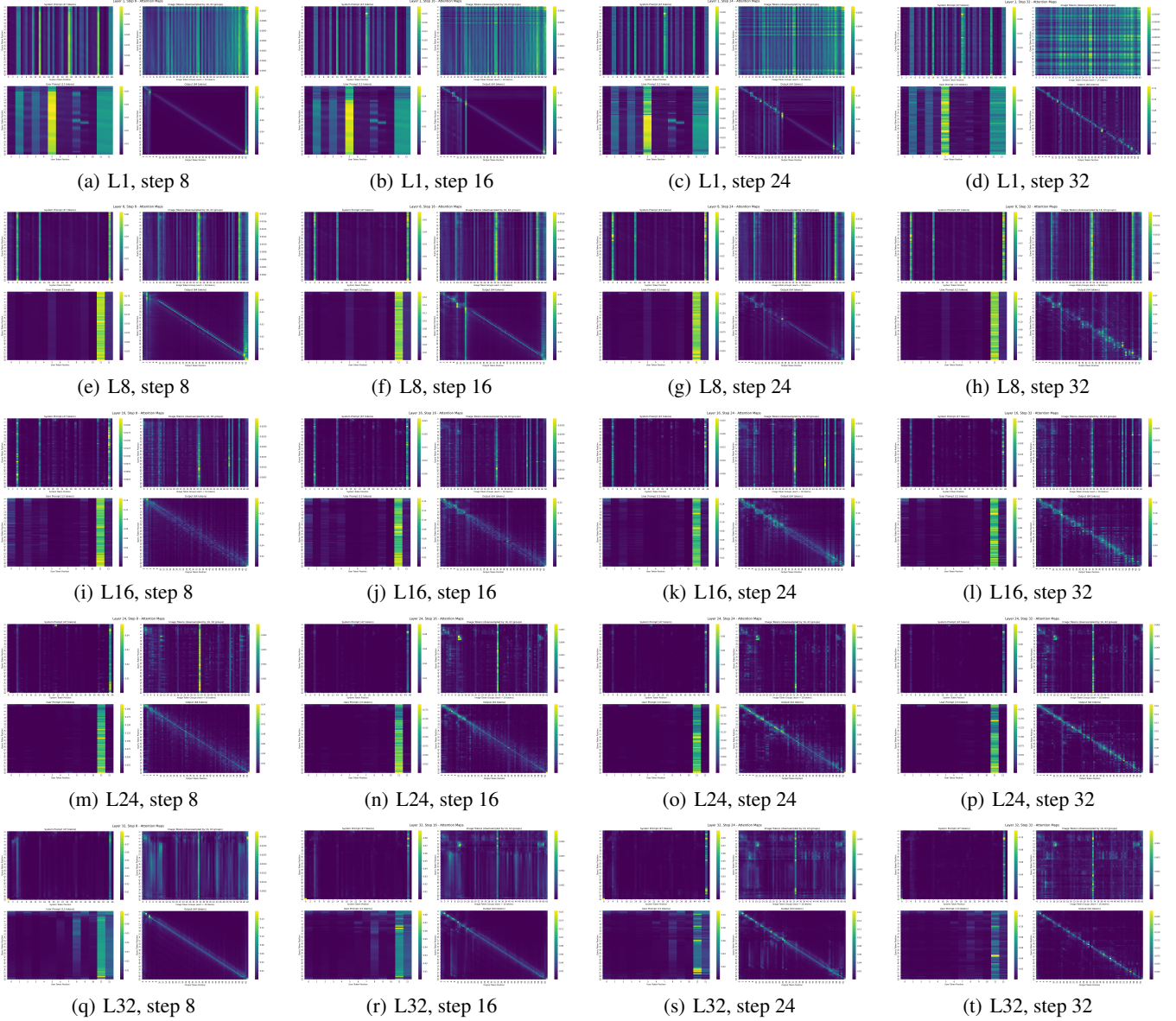


Figure 9: Query-key attention matrices sampled across depth and time. Dark colours indicate low weights. From left to right the temporal focus sharpens; from top to bottom the spatial focus narrows. The intersection of these trends visualises how the two-level dynamic merge schedule first preserves global context then concentrates computation on the salient image groups identified by the decider tokens.

Figure 8(c) shows that, for a given retention level, all merge algorithms yield comparable theoretical FLOP reductions; differences arise from implementation overhead and are reflected in wall-clock time. Figure 8(d) corroborates this: D³ToM-t lowers latency from 786.4 s at 50% tokens to 493.9 s at 10%, a $2.34\times$ speed-up that closely tracks the FLOP savings.

Taken together, the plots visually substantiate the numerical claim that timetable-aware merging (D³ToM-t) provides the best cost-effectiveness: up to 69.8% compute saved and $2.3\times$ faster inference while preserving at least 96.7% of

baseline accuracy on average.

G Broader Impacts

Our work introduces D³ToM, the first method to accelerate diffusion-based MLLMs by exploiting visual-token redundancy directly within the iterative denoising loop. The core innovation is a dual-dynamic mechanism: the token merge pattern adapts at each step based on the model’s evolving attention, while the merge ratio follows a coarse-to-fine schedule.

By reducing computational requirements by up to 70%



Figure 10: Visualization of the most important visual token group at each denoising step (Part 1). The highlighted region in each subfigure corresponds to the visual token with the maximum decoder-guided importance score.

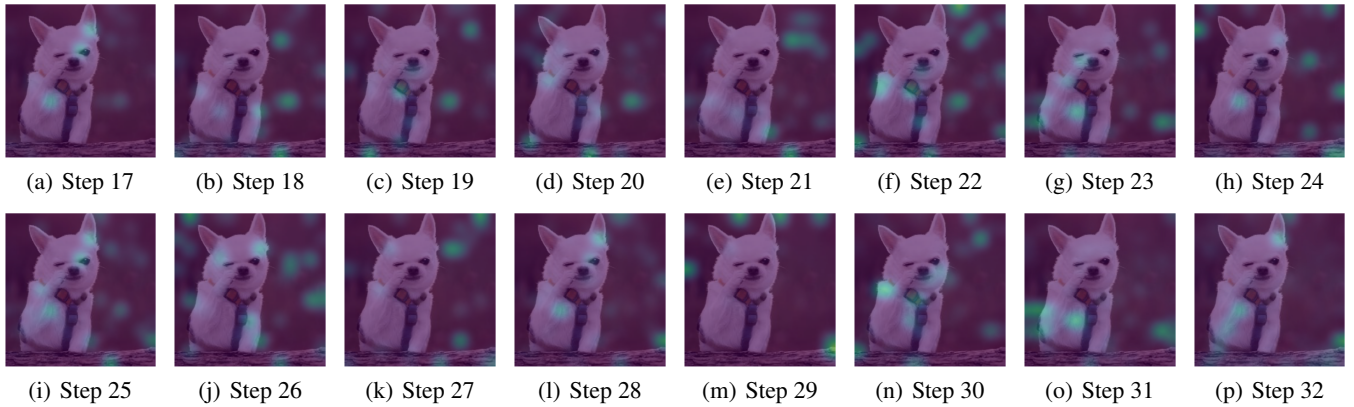


Figure 10: Visualization of the most important visual token group at each denoising step (Part 2). The shifting focus demonstrates the dynamic nature of visual salience during the iterative generation process.

with minimal impact on accuracy, D³ToM significantly lowers the barrier to entry for deploying state-of-the-art diffusion MLLMs. This enhances sustainability by reducing the energy cost per query and democratizes access for researchers and organizations with limited computational resources.

Our current validation is focused on image understanding benchmarks due to the limitation of modes. Besides, we use a single diffusion language model backbone LLaDA-8B architecture. Promising future research directions include extending the future diffusion-based video MLLMs and curiosity language model backbone such as Dream-7B. Critically, future work should also investigate methods to mitigate the risk of bias amplification. This could involve developing fairness-aware merge schedules or conducting detailed attribution analyses on demographically balanced datasets to ensure that information critical to equity is preserved during the merging process.

H Additional Visualizations

H.1 Attention dynamics during diffusion decoding

Figure 9 visualizes the evolution of attention patterns, revealing a dual trend of specialization across both time and network depth. Temporally (left to right), attention refines from a diffuse state at early denoising steps to a highly sparse pattern by the final steps. Spatially (top to bottom), this sparsity is most pronounced in deeper layers (e.g., L24, L32), which concentrate computation on a small subset of visual tokens, while shallower layers retain a broader, more contextual focus.

These evolving patterns are a direct consequence of our decoder-guided mechanism. The bright vertical bands in the ‘Output-to-Image’ attention (lower-right quadrant) show newly generated “decoder” tokens focusing on salient visual information. This attention pattern is used to score and prune the visual token set. The resulting sparsity in the ‘Image-to-Image’ attention (upper-right quadrant) is the direct effect of this pruning, as merged tokens are removed from the compu-

tation. This provides clear visual evidence that our method successfully reallocates computational resources in the visual domain based on the dynamically generated text.

H.2 Dynamic Decider-Guided Importance Peaks

Our decider-guided mechanism computes an importance score for each visual token group at every denoising step. Figure 10 visualizes the evolution of these scores by highlighting the visual patch that receives the maximum importance score at each of the 32 steps. The visualization clearly demonstrates that visual salience is dynamic; as the model refines its generated response from an initial coarse draft to a final polished output, the focus of its attention shifts across different regions of the input image. For full reproducibility, the complete set of importance scores for all groups at every step is provided in the supplementary materials.