LOOK ONLY WHEN NEEDED: CERTIFICATE-GUIDED VISUAL ROUTING FOR EFFICIENT DIFFUSION VLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion vision-language models (DVLMs) process their entire input at every denoising step. Since visual tokens make up 89–97% of this input on the benchmarks we study, most of the decoding computation is spent repeatedly processing the same image. Visual token pruning methods partially relieve this burden by removing part of the visual tokens, but they lose the removed information and still process the remaining image at every step. To measure how often the image is actually needed, we compare LLaDA-V's predictions with and without the image at every decoding step and find that removing the image changes a token being unmasked at only 22.8% of steps on average and at as few as 7.9%. We therefore propose LOWN, a decoding method that decides at every step whether to process the image and how many tokens to unmask. LOWN predicts the masked tokens without the image and processes the image only when this prediction disagrees with the last image-conditioned prediction. Agreement certifies the earlier prediction for reuse, and the number of certified tokens sets how many are unmasked in that step. We further observe that the need for the image differs across benchmarks, so LOWN's adaptive decision is needed rather than a fixed schedule. Without training or a tuned threshold, LOWN retains 99.6% of LLaDA-V's score on ten benchmarks at 35.3% of the decoding FLOPs on average. LOWN also generalizes across LLaDA-based DVLMs, composes orthogonally with visual token pruning, and runs under key-value caching for further efficiency gains.

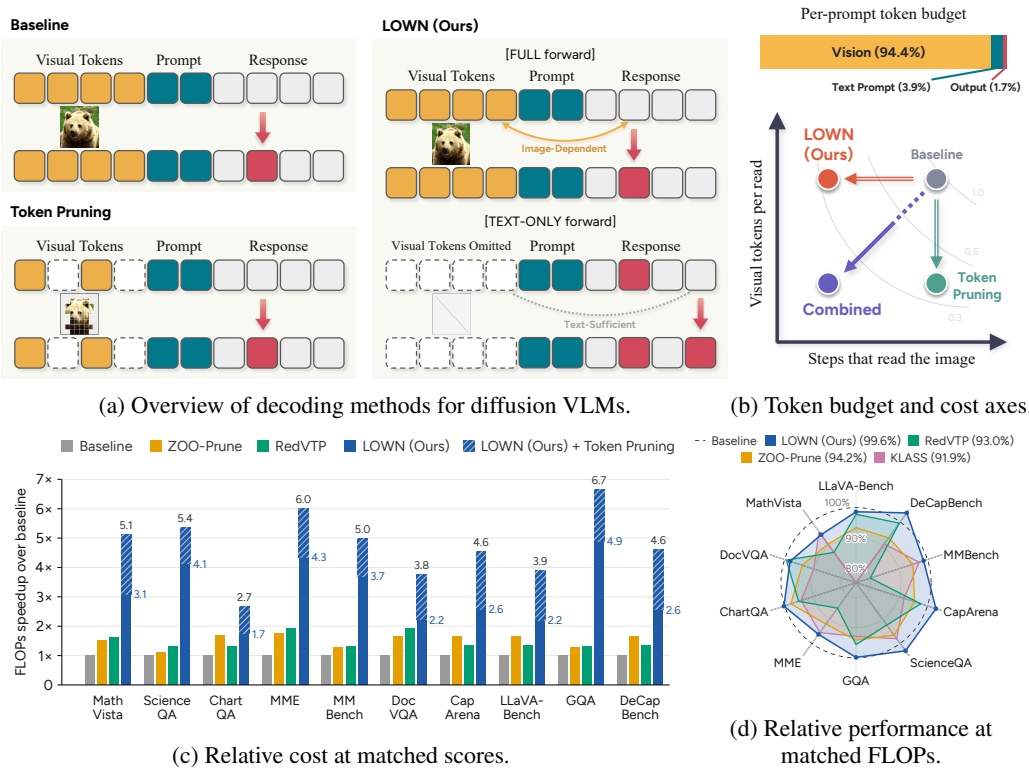
1 INTRODUCTION

Recent diffusion language models match similarly sized autoregressive models while refining several masked positions in parallel over denoising steps (Nie et al., 2025; Ye et al., 2025a). In diffusion vision-language models (DVLMs) such as LLaDA-V (You et al., 2026) and others (Yang et al., 2025a; Yu et al., 2025), bidirectional attention prevents direct use of the exact key-value caching found in autoregressive models, so standard inference processes the full multimodal sequence at every step. Visual tokens dominate this sequence, accounting for 89–97% of LLaDA-V's input across our benchmarks (Figure 1b), while a forward without them costs only 3–10% as much. DVLM decoding therefore spends most of its compute repeatedly processing an unchanged visual input.

Existing methods reduce this cost through visual token pruning (Chang et al., 2026; Xu et al., 2026; Kim et al., 2026) (Figure 1a), intermediate-state caching (Wu et al., 2026a; Liu et al., 2026; Nguyen-Tri et al., 2026), or unmasking more tokens per step (Kim et al., 2025; Li et al., 2026b; Wu et al., 2026b; Wang et al., 2026; Lee et al., 2026). These approaches make image-conditioned forwards cheaper, reuse their computation, or run fewer decoding steps, but none asks whether the current step needs the image. We ask whether removing the image would change the model's decision at the current decoding state.

Comparing LLaDA-V's predictions with and without the image along the same decoding trajectory reveals substantial redundancy (Section 3.2). The two give the same top-1 token at 83.4% of the positions selected for unmasking, and 85.0% of final tokens had already appeared as top-1 predictions before unmasking. Within a trajectory, image-dependent steps tend to cluster early rather than recur periodically, while their overall share varies from 7.9% on MMBench to 43.8% on DocVQA. A fixed schedule therefore finds it hard to capture when the image is needed. These observations motivate deciding image access per input and step from predictions the model has already computed.

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.



(a) Overview of decoding methods for diffusion VLMs.

(b) Token budget and cost axes.

(c) Relative cost at matched scores.

(d) Relative performance at matched FLOPs.

Figure 1: **Look Only When Needed (LOWN).** (a) Decoding with a diffusion VLM. Token pruning reduces the visual tokens processed during decoding, whereas LOWN decides at each step whether the visual tokens need to be processed at all and otherwise runs a TEXT-ONLY forward. (b) Composition of the input, averaged over ten benchmarks. Pruning reduces how many visual tokens each forward processes, LOWN whether a forward processes them at all. (c) Decoding FLOPs speedup. Each pruning method is shown at LOWN's score (Appendix C.3), and LOWN + Token Pruning uses RedVTP (r=50%). (d) Quality relative to LLaDA-V when every method is run at the operating point closest to LOWN's decoding FLOPs from above. LOWN keeps 99.6% on average.

We propose LOWN (Look Only When Needed), a decoding method built on this redundancy. LOWN stores the top-1 predictions and confidences from the latest image-conditioned forward, which we call the FULL forward, as an anchor and tests it with a cheap TEXT-ONLY forward at each step. Agreement certifies the anchor prediction for reuse, while disagreement triggers a new FULL forward. The confidence-ordered run of agreements also determines how many tokens to unmask. LOWN requires neither training nor a task-specific threshold and can be added to visual token pruning or key-value caching because it reduces how often a FULL forward is needed (Figure 1b).

We evaluate LOWN on ten benchmarks spanning multiple-choice, document and chart understanding, mathematical reasoning, and open-ended captioning. On LLaDA-V, it retains 99.6% of the baseline score at 0.35× the decoding FLOPs (Figures 1c and 1d). Against accelerated methods given at least its FLOPs budget, it scores highest on eight of the ten benchmarks. Combined with key-value caching or visual token pruning, it further cuts the partner's FLOPs by 1.8–2.5×, and with the key-value cache the pair decodes 5.6–7.0× faster than the model alone. The same rule transfers to LaViDa and VidLaDA.

Our contributions are as follows.

- We show through paired FULL and TEXT-ONLY forward comparisons that most DVLM decoding steps can skip image access, while the necessary steps vary across trajectories and are not well captured by fixed schedules.
- We introduce LOWN, a training-free and threshold-free method that uses prediction agreement to jointly decide when to process the image and how many tokens to unmask.
- We validate LOWN across ten benchmarks and three DVLM backbones, retaining 99.6% of LLaDA-V's score at 0.35× its decoding FLOPs, composing orthogonally with visual token pruning, and running under key-value caching.

2 Related Work

Diffusion language and vision-language models. Discrete diffusion and masked diffusion models generate sequences through iterative token recovery (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024). This paradigm has scaled to large language models (Nie et al., 2025; Ye et al., 2025a) and extended to image and video understanding (DiffusionGemma Team, 2026; You et al., 2026; Li et al., 2025; Yu et al., 2025; He et al., 2026). These models establish the DVLM setting studied here. LOWN addresses the complementary inference question of how often denoising must reprocess the visual input.

Efficient inference for diffusion LMs and VLMs. Visual-token reduction shortens the visual sequence in vision transformers and autoregressive VLMs (Bolya et al., 2023; Chen et al., 2024; Zhang et al., 2025; Xing et al., 2025; Shang et al., 2025; Yang et al., 2025b). ZOO-Prune (Kim et al., 2026) combines projector-level sensitivity estimates with diversity-aware selection.

For diffusion VLMs, RedVTP prunes using attention from still-masked response tokens, while D^{3}ToM uses previously decoded tokens to guide visual-token selection and merging (Xu et al., 2026; Chang et al., 2026). These methods determine which visual tokens a forward processes, but they lose the removed information (Endo et al., 2025; Li et al., 2026a).

Other work keeps the visual input and reduces the number of steps or the cost of a forward. Decoding schedules unmask multiple tokens per step when confidence, cross-step stability, or attention to the image supports it (Wei et al., 2026; Wu et al., 2026a; Yu et al., 2025; Wu et al., 2026b; Wang et al., 2026; Lee et al., 2026). Prophet (Li et al., 2026b) unmasks all remaining tokens once the average top-two logit gap in the answer region is large enough. KLASS (Kim et al., 2025) combines confidence with cross-step distribution stability and is our schedule baseline. Caching methods reuse intermediate states across denoising steps (Ma et al., 2025; Liu et al., 2026; Nguyen-Tri et al., 2026; Li et al., 2025), and we compose LOWN with Fast-dLLM (Wu et al., 2026a), a blockwise approximate key-value cache. Pruning and caching act inside a forward, on the tokens it processes or the states it reuses, and schedules on how many tokens a step unmasks. LOWN decides both whether a step processes the image and how many tokens it unmasks from the agreement between a TEXT-ONLY and the last image-conditioned prediction, and, keeping only predictions, it runs on top of pruning and caching.

3 Background and Motivation

3.1 Preliminaries

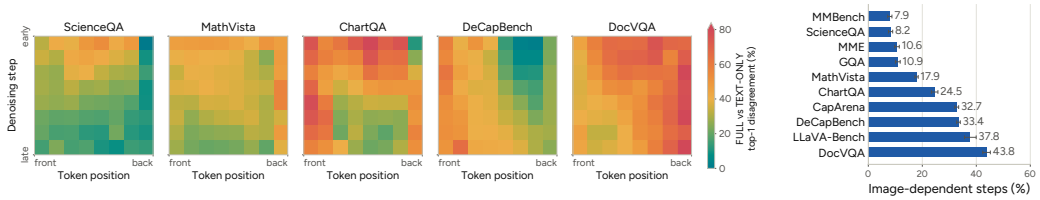
Starting from a fully masked response, a DVLM repeatedly predicts response tokens over T denoising steps while conditioning on fixed visual and prompt tokens (Sahoo et al., 2024; Nie et al., 2025; You et al., 2026). Let v denote the visual tokens, c the prompt tokens, y the response, x_t the partially masked response at step t, and M_t its masked positions. A *FULL forward* over (v, c, x_t) gives the top-1 prediction and confidence at each $i \in M_t$,

$$\hat{y}^{\text{full}}_{t,i} = \arg\max_w p_\theta(y^i{=}w \mid v, c, x_t), \qquad \hat{s}^{\text{full}}_{t,i} = \max_w p_\theta(y^i{=}w \mid v, c, x_t). \tag{1}$$

LLaDA-V runs a FULL forward at every step and unmasks the k most confident positions (Nie et al., 2025; You et al., 2026), where k is the response length divided by T.

3.2 When Does a DVLM Need the Image?

To measure how often the DVLM uses the image, we compare two forward passes along LLaDA-V's standard decoding trajectory on all ten benchmarks of Section 5. The FULL forward evaluates $p_\theta(\cdot \mid v, c, x_t)$, whereas the *TEXT-ONLY forward* removes the visual tokens while preserving the position IDs and evaluates $p_\theta(\cdot \mid c, x_t)$. At every $i \in M_t$, we record the top-1 predictions from both forwards, $\hat{y}^{\text{full}}_{t,i}$ and $\hat{y}^{\text{text}}_{t,i}$, together with the FULL confidence $\hat{s}^{\text{full}}_{t,i}$. The FULL forward alone determines the decoding trajectory, and the TEXT-ONLY outputs are recorded without affecting its decisions. Appendix A gives the full protocol and per-benchmark values.



(a) Where and when the image changes a prediction. (b) Image-dependent steps.

Figure 2: **Visual dependence along the decoding trajectory of LLaDA-V.** (a) Share of masked positions at which the FULL and TEXT-ONLY top-1 predictions disagree, over token position (front to back) and denoising step (early to late). (b) Share of steps on which the image changed at least one unmasked decision, per benchmark, with 95% confidence intervals.

The image rarely changes a decision. Of the positions the model unmasks, 83.4% receive the same top-1 token from the FULL and the TEXT-ONLY forward, averaged over the ten benchmarks, so at most steps the prediction follows from the text alone and the presence of the visual tokens rarely changes the token being unmasked. Moreover, 85.0% of the final tokens had already appeared as the top-1 prediction at the same position before the step at which they were unmasked. The baseline nevertheless continued to recompute those positions with the image until unmasking them.

The need for image access varies along the decoding trajectory. On eight of the ten benchmarks, disagreement between the two forwards is largest in the early denoising steps and decreases as the response takes shape. This pattern is not universal, as illustrated by DeCapBench in Figure 2a. The affected token positions also differ across tasks, and the share of image-dependent steps ranges from 7.9% on MMBench to 43.8% on DocVQA, 22.8% on average (Figure 2b). Variation across steps, positions, and benchmarks therefore makes a fixed image-access schedule poorly matched to the observed trajectories and motivates input- and step-adaptive image access.

A method that skips the image therefore needs a test that runs at every step, and the TEXT-ONLY forward that exposed the redundancy is a natural candidate, since it costs 3–10% of a FULL forward. The next section turns the agreement between the two forwards into that test.

4 METHOD

LOWN separates two decisions that standard DVLM decoding makes with a single FULL forward, whether the current response state needs the image and how far decoding can advance (Figure 3). It retains the last image-conditioned prediction as an anchor, tests that anchor with a TEXT-ONLY forward, and runs a FULL forward only when the test cannot certify enough tokens. The same test controls how many tokens are unmasked, allowing a successful TEXT-ONLY step to advance farther than the baseline. LOWN changes neither the model parameters nor the attention pattern.

4.1 ANCHOR AND CERTIFICATE

Whenever LOWN runs a FULL forward, it retains its predictions $\bar{y}$ and confidences $\bar{s}$ on the positions that remain masked, and we call this pair the *anchor*. At step t, LOWN runs a TEXT-ONLY forward and orders the masked positions by decreasing anchor confidence $\bar{s}$. A candidate token i is *certified* when its current TEXT-ONLY prediction agrees with the anchor, $\hat{y}_{t,i}^{\text{text}} = \bar{y}_i$. The two predictions differ in whether they see the image and in the response state they were computed from, so their agreement approximates two conditions at once, that the image is not needed for this token and that the anchor remains valid after the state has changed, without an absolute confidence threshold. In the measurement of Section 3.2, a fresh FULL forward overturns 4.1–11.0% of the tokens certified against a one-step-old anchor, and in LOWN's own runs it overturns 2.2–4.7% of the tokens unmasked on TEXT-ONLY steps (Section 5.3).

The longest confidence-ordered prefix of certified candidates forms $\mathcal{U}^{\text{text}}$. The first disagreement ends the prefix, so the acceptance length $|\mathcal{U}^{\text{text}}|$ is set by the current input and step rather than by a constant. The scan also applies a repetition guard, detailed in Appendix B.1. The certificate is checked against a prediction that ages as the response changes, and its error grows with that age (Section 5.3). LOWN therefore replaces the entire anchor whenever it runs a FULL forward, so that every certificate is checked against the most recent FULL forward.

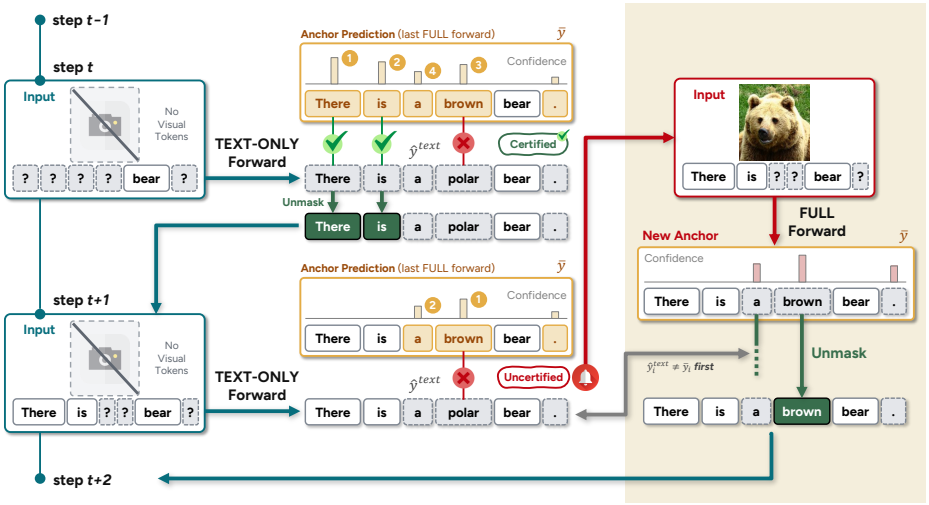


Figure 3: **Overview of LOWN.** TEXT-ONLY agreement certifies anchor tokens for image-free unmasking, and an insufficient certified prefix triggers a FULL forward that refreshes the anchor and unmasks the positions where the two forwards disagree first.

4.2 Certificate-Guided Decoding

Each step runs a TEXT-ONLY forward and forms the certified prefix $\mathcal{U}^{\text{text}}$ of Section 4.1. The length of this prefix decides whether the step also needs a FULL forward. When the prefix contains at least k tokens, where k is the number of tokens the baseline unmasks per step, the step skips the FULL forward and accepts the whole prefix, unmasking every certified position to its anchor prediction rather than only k of them. This whole-prefix acceptance is what lets one step unmask more than k tokens. When the prefix is shorter than k, the anchor no longer certifies enough tokens for a step, so the step runs a FULL forward, replaces the anchor with its predictions and confidences, and unmasks k positions to their FULL predictions in disagreement-first order. Positions whose TEXT-ONLY and FULL predictions differ, $\hat{y}^{\text{text}}_{t,i} \neq \hat{y}^{\text{full}}_{t,i}$, come first, and the remaining positions follow FULL confidence. These k positions form $\mathcal{U}^{\text{full}}$.

The resulting state update is

$$x_i \leftarrow \begin{cases} \bar{y}_i, & i \in \mathcal{U}^{\text{text}}, \quad |\mathcal{U}^{\text{text}}| \geq k, \\ \hat{y}^{\text{full}}_{t,i}, & i \in \mathcal{U}^{\text{full}}, \quad |\mathcal{U}^{\text{text}}| < k. \end{cases} \tag{2}$$

Both branches unmask at least k tokens, so LOWN never takes more than the baseline's T steps, while the TEXT-ONLY branch can take fewer. Appendix B.1 provides the detailed algorithm.

4.3 Cost Analysis

Let $\mathcal{C}(n)$ denote the FLOPs of one forward over n tokens. The FULL and TEXT-ONLY input lengths are $N_F = |v| + |c| + |x_t|$ and $N_T = |c| + |x_t|$. If the baseline runs T FULL forwards, its cost is $F_{\text{base}} = T \cdot \mathcal{C}(N_F)$. LOWN runs n_T TEXT-ONLY forwards and adds a FULL forward on n_F of those steps, which gives $F_{\text{LOWN}} = n_F \cdot \mathcal{C}(N_F) + n_T \cdot \mathcal{C}(N_T)$. Relative FLOPs are therefore

$$R = \frac{F_{\text{LOWN}}}{F_{\text{base}}} = \frac{n_F}{T} + \frac{n_T}{T}\frac{\mathcal{C}(N_T)}{\mathcal{C}(N_F)} = f + \sigma\rho. \tag{3}$$

Here $f = n_F/T$ is the frequency of FULL forwards, $\sigma = n_T/T \leq 1$ is the relative number of steps, and $\rho = \mathcal{C}(N_T)/\mathcal{C}(N_F)$ is the unit cost of a TEXT-ONLY forward. The certificate lowers f, and accepting more than k tokens can lower σ. The input determines ρ, which decreases as visual tokens occupy more of the FULL input. Appendix B gives $\mathcal{C}$ and the per-benchmark ρ.

Visual token pruning changes only the FULL term. Let $\pi = \mathcal{C}(N_F^{\text{pruned}})/\mathcal{C}(N_F)$ be the FULL-forward cost with the pruned visual input. Pruning alone costs π relative to the baseline, while its composition with LOWN costs $\pi f + \sigma\rho$, and both are charged the pruning overhead once. Pruning reduces the cost of each FULL forward, whereas LOWN reduces how often one is run. Section 5 evaluates this composition.

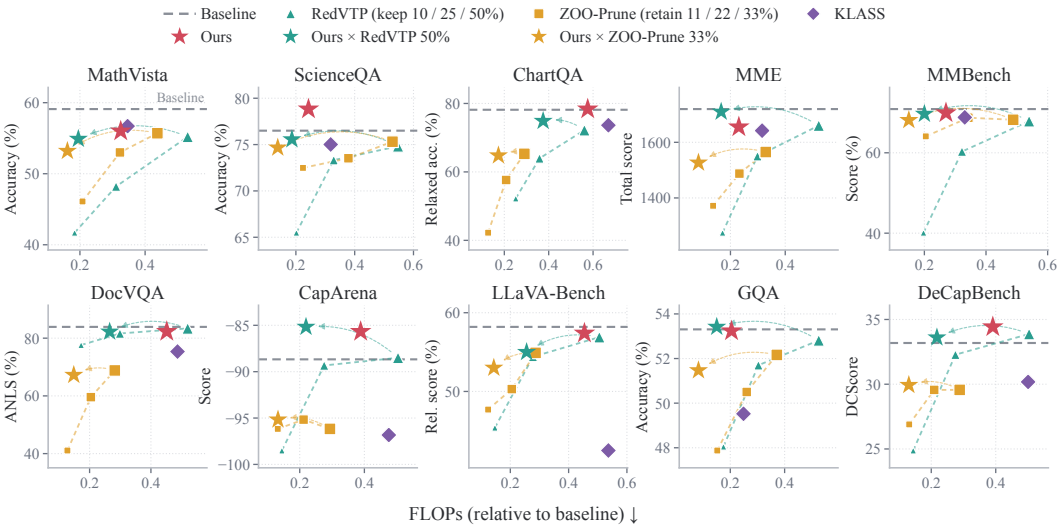


Figure 4: **Performance and efficiency on LLaDA-V**. Gray dashed lines denote the baseline. Triangles and squares denote RedVTP and ZOO-Prune retention settings, respectively. Arrows connect each fixed-pruning setting to its composition with LOWN. KLASS is matched to LOWN's FLOPs.

5 Experiments

Our experiments test whether LOWN preserves quality while reducing decoding cost. Section 5.1 defines the models, benchmarks, baselines, and evaluation protocol. Section 5.2 evaluates standalone LOWN, its composition with visual token pruning and key-value caching, and its transfer to other image and video DVLMs. Section 5.3 compares image-access schedules, isolates visual routing and adaptive acceptance, and analyzes visual information preservation and certificate reliability.

5.1 Implementation Details

Benchmarks and evaluation. We evaluate LOWN on LLaDA-V (You et al., 2026) and LaViDa (Li et al., 2025) across ten benchmarks, namely MathVista (Lu et al., 2024), ScienceQA (Lu et al., 2022), ChartQA (Masry et al., 2022), MME (Fu et al., 2025), MMBench (Liu et al., 2024), DocVQA (Mathew et al., 2021), GQA (Hudson & Manning, 2019), LLaVA-Bench (Liu et al., 2023), CapArena (Cheng et al., 2025), and DeCapBench (Ye et al., 2025b). We additionally evaluate VidLaDA-8B (He et al., 2026) on EgoSchema (Mangalam et al., 2023) and MLVU (Zhou et al., 2025). We evaluate quality using each benchmark's task-specific metric and apply the same scorer to all methods within a comparison. Analytical decoding FLOPs are reported relative to each backbone's baseline.

Baselines. We compare RedVTP (Xu et al., 2026), ZOO-Prune (Kim et al., 2026), KLASS (Kim et al., 2025), and D^{3}ToM (Chang et al., 2026). All methods share prompts and response budgets within each comparison. Where multiple operating points are available, we select settings by FLOPs rather than task scores and assign each comparator at least the corresponding LOWN budget. We apply the same principle to LLaDA-V and LaViDa and evaluate Fast-dLLM (Wu et al., 2026a) separately in the caching comparison.

LOWN implementation. For standalone LOWN on LLaDA-V, we use batch size 1 and greedy decoding with temperature and classifier-free guidance scale set to zero, without key-value caching. The image is encoded once. FULL forwards reuse its visual embeddings, whereas TEXT-ONLY forwards remove those embeddings while preserving the original position IDs of the retained prompt and response tokens. Appendix C provides the complete evaluation protocol, baseline settings, and decoding details.

Table 1: **Performance at matched decoding FLOPs on LLaDA-V**. The upper block compares standalone methods at LOWN's budget, and the lower block at the budget of LOWN with RedVTP (r=50%). Competitors run on a discrete hyperparameter grid, so each cell uses the cheapest setting with at least LOWN's FLOPs (Appendix C.3). Cells show scores with FLOPs relative to LLaDA-V below. Best scores are bold, second best underlined, and lowest FLOPs bold.

Method	Math Vista	Science QA	Chart QA	MME	MM Bench	Doc VQA	Cap Arena	LLaVA-Bench	GQA	DeCap Bench	Avg.
LLaDA-V *(Vanilla)*	59.10	76.50	78.16	1719.49	70.88	83.91	−88.67	58.20	53.31	33.18	100.0%
FLOPs matched to LOWN											
RedVTP	51.70 (×0.354)	70.70 (×0.286)	74.32 (×0.641)	1469.78 (×0.256)	56.70 (×0.280)	**83.39** (×0.520)	−91.00 (×0.411)	56.90 (×0.505)	50.97 (×0.260)	33.03 (×0.411)	93.0% (×0.392)
ZOO-Prune	53.70 (×0.403)	73.87 (×0.348)	76.44 (×0.641)	1582.41 (×0.299)	67.27 (×0.317)	78.73 (×0.483)	−93.17 (×0.417)	54.30 (×0.490)	50.16 (×0.238)	31.02 (×0.411)	94.2% (×0.405)
KLASS	**56.70** (×0.346)	75.01 (×0.318)	73.64 (×0.669)	1641.99 (×0.316)	68.81 (×0.331)	75.36 (×0.487)	−96.83 (×0.476)	42.50 (×0.536)	49.52 (×0.249)	30.18 (×0.500)	91.9% (×0.423)
LOWN (Ours)	56.00 **(×0.325)**	**78.83** **(×0.243)**	**78.36** **(×0.576)**	**1655.08** **(×0.231)**	**69.93** **(×0.270)**	82.31 **(×0.452)**	**−85.67** **(×0.388)**	**57.40** **(×0.454)**	**53.24** **(×0.205)**	**34.43** **(×0.391)**	**99.6%** **(×0.353)**
FLOPs matched to LOWN (Ours) + RedVTP (r=50%)											
RedVTP	48.40 (×0.268)	65.49 (×0.202)	66.00 (×0.400)	1275.08 (×0.171)	56.70 (×0.280)	81.62 (×0.299)	−91.33 (×0.232)	54.40 (×0.278)	48.05 (×0.175)	31.16 (×0.231)	87.8% (×0.253)
ZOO-Prune	46.10 (×0.208)	72.48 (×0.224)	71.68 (×0.418)	1487.83 (×0.214)	64.09 (×0.205)	68.82 (×0.282)	−95.50 (×0.268)	54.90 (×0.287)	47.88 (×0.152)	29.12 (×0.265)	88.8% (×0.252)
KLASS + RedVTP *(r=50%)*	52.70 (×0.207)	73.48 (×0.210)	68.92 (×0.428)	1584.56 (×0.191)	65.29 (×0.207)	76.65 (×0.274)	−96.67 (×0.247)	42.50 (×0.290)	48.93 (×0.154)	29.02 (×0.258)	89.3% (×0.247)
LOWN (Ours) + RedVTP *(r=50%)*	**54.90** **(×0.195)**	**75.56** **(×0.186)**	**74.88** **(×0.375)**	**1709.75** **(×0.166)**	**69.67** **(×0.200)**	**82.24** **(×0.265)**	**−85.17** **(×0.219)**	**55.00** **(×0.256)**	**53.42** **(×0.150)**	**33.61** **(×0.218)**	**98.3%** **(×0.223)**

5.2 Main Results

Results on LLaDA-V. The upper block of Table 1 reports LOWN's standalone results. LOWN retains 99.6% of the baseline score on average at 0.353× the baseline FLOPs. It scores highest among the accelerated methods on eight of ten benchmarks. LOWN uses one adaptive rule without additional tuning, whereas KLASS reports benchmark-specific thresholds. The 99.6% average includes four benchmarks on which LOWN scores above the baseline, and capping each benchmark at 100% so that these gains do not offset losses elsewhere gives 98.6%. Visual routing alone exceeds the baseline on ScienceQA and CapArena, and disagreement-first order lifts ChartQA and DeCapBench above it, raising the score on seven of the ten benchmarks (Table 14). On a FULL step, this order first unmasks the positions where the image changed the prediction, so image-dependent tokens are not deferred.

Composition with visual token pruning. We combine LOWN with RedVTP and ZOO-Prune without retraining or changing their retention ratios. With RedVTP at 50% retention, LOWN reaches a normalized average of 98.3 at 0.223× the baseline FLOPs and scores highest on all ten benchmarks (Table 1). It further reduces FLOPs by 57.6% for RedVTP at 50% retention and 60.6% for ZOO-Prune at 33% retention (Table 11). For a visual-token pruner, further cost reductions require discarding more visual information. LOWN instead reduces how often the retained visual tokens are processed, allowing less aggressive pruning to outperform a more aggressive setting in both quality and compute. On MathVista, RedVTP at 50% retention with LOWN scores 54.9 at 0.195× FLOPs, against 51.7 at 0.354× for RedVTP at 30% retention. Figure 4 shows the broader trade-off.

Table 2: Composition of LOWN and Fast-dLLM on LLaDA-V.

Method	MathVista			LLaVA-Bench			DeCapBench		
	Score ↑	Time (s) ↓	FLOPs ↓	Score ↑	Time (s) ↓	FLOPs ↓	Score ↑	Time (s) ↓	FLOPs ↓
LLaDA-V	59.10	7.56	1.0000	58.20	6.99	1.0000	33.18	14.64	1.0000
Fast-dLLM	57.30	2.14	0.0856	63.00	2.02	0.0876	35.19	4.22	0.1104
Fast-dLLM + LOWN	57.10	1.23 (−42.4%)	0.0457 (−46.6%)	58.00	1.25 (−38.1%)	0.0489 (−44.1%)	33.35	2.10 (−50.1%)	0.0453 (−58.9%)

Composition with key-value caching. LOWN also provides plug-and-play acceleration on top of key-value caching, since the cache lowers the cost of each forward and LOWN's acceptance lowers the number of forwards. We retain Fast-dLLM's cache block and refresh settings and combine its token selection with LOWN's acceptance rule. Across MathVista, LLaVA-Bench, and DeCapBench, adding LOWN reduces mean generation time by 38.1–50.1% (1.61–2.01× further speedup) and

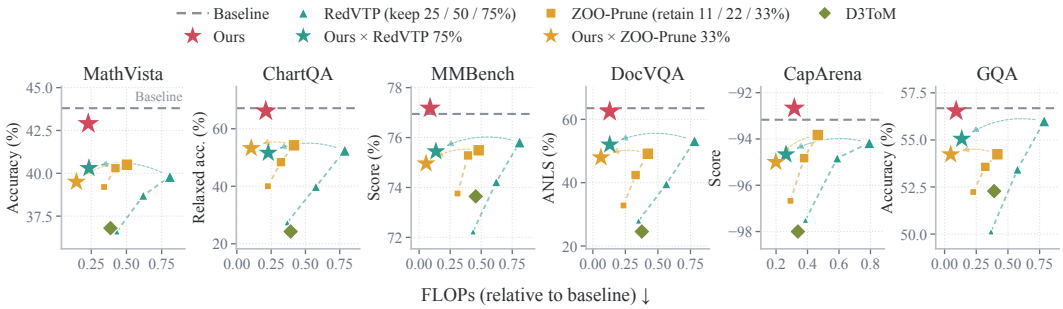


Figure 5: **Performance and efficiency on LaViDa**. Gray dashed lines denote the baseline. Triangles, squares, and diamonds denote RedVTP, ZOO-Prune, and D^{3}ToM, respectively. Composition markers use RedVTP at 75% and ZOO-Prune at 33% retention.

FLOPs by 44.1–58.9% relative to Fast-dLLM (Table 2, Appendix D). The combined scores are within 2.0 points of LLaDA-V, but lower than Fast-dLLM on all three tasks, with a decrease of 5.0 points on LLaVA-Bench.

Other models. *LaViDa.* LOWN transfers to LaViDa without changing its decision rule, retaining 99.7% of the baseline score on average at 0.174× its decoding FLOPs (Figure 5). On MMBench and GQA, LOWN scores 77.18 versus 76.95 and 56.54 versus 56.69 at only 0.088× and 0.091× the FLOPs. As on LLaDA-V, adding LOWN to RedVTP at 75% or ZOO-Prune at 33% retention cuts their FLOPs by a further 77.3% and 72.2% at quality indices of 99.5 and 99.8. These results show that adaptive image access transfers across DVLM backbones and remains complementary to visual token pruning. Appendix E reports the complete results.

VidLaDA. We apply LOWN to VidLaDA-8B on two video benchmarks under a controlled 128-token decoding protocol (Table 3, Appendix C.4). LOWN uses 0.255× the baseline decoding FLOPs on EgoSchema and 0.272× on MLVU, with scores of 60.2 and 43.3 compared with the baseline's 60.6 and 41.8.

Table 3: Video-understanding results of LOWN on VidLaDA-8B.

	EgoSchema		MLVU	
Method	Score ↑	FLOPs ↓	Score ↑	FLOPs ↓
VidLaDA	60.6	1.000	41.8	1.000
LOWN (Ours)	60.2	0.255 (−74.5%)	43.3	0.272 (−72.8%)

Wall-clock latency. Across the ten LLaDA-V benchmarks, standalone LOWN reduces generation time by 57.4% on average and up to 73.2%. Appendix D repeats Table 1 with each competitor given at least LOWN's generation time, for standalone methods and pruning compositions.

5.3 Ablation and Analysis

Image-access scheduling. Table 4 compares LOWN with periodic and random image access. Period n uses the image every n steps, while Random matches LOWN's average FULL-forward count. Against this matched random schedule, LOWN improves task-normalized quality by 4.9–13.0% on all three tasks. Period 4 uses similar compute to LOWN on MathVista but scores 4.0 points lower.

Table 4: Comparison of LOWN with fixed and random image-access schedules on LLaDA-V. Cells report score and relative FLOPs.

	MathVista	LLaVA-Bench	DeCapBench
LOWN (Ours)	56.00 (×0.325)	57.40 (×0.454)	34.43 (×0.391)
Period 2	57.00 (×0.541)	56.10 (×0.516)	33.43 (×0.522)
Period 4	52.00 (×0.321)	49.00 (×0.274)	27.79 (×0.284)
Period 8	48.50 (×0.211)	41.30 (×0.154)	19.09 (×0.164)
Random	53.40 (×0.349)	53.80 (×0.456)	30.48 (×0.389)

Visual routing and adaptive acceptance. Table 5 separates two decisions of LOWN on LLaDA-V. *Visual routing* is the certificate's choice of the steps that run a FULL forward, and *adaptive acceptance* is whole-prefix acceptance on TEXT-ONLY steps together with disagreement-first order on FULL steps (Section 4.2, Appendix F.1). Visual routing alone, which keeps the baseline's order and unmasks k tokens per step, reduces FLOPs by 27–68%. Adaptive acceptance removes a further 21–43% of the remaining cost, scores higher on seven of the ten benchmarks, and loses at most 2.8% of normalized quality on the others. Most of the saving therefore comes from skipping FULL forwards, and adaptive acceptance adds a task-dependent reduction. This is not the same as running fewer steps, since we find that uniformly using two or four times fewer steps degrades performance across benchmarks (Table 12).

Table 5: Ablation of visual routing and adaptive acceptance on LLaDA-V. Each setting reports scores and, below, relative FLOPs with the reduction from the preceding setting in parentheses.

Visual routing	Adaptive acceptance	Math Vista	Science QA	Chart QA	MME	MM Bench	Doc VQA	Cap Arena	LLaVA-Bench	GQA	DeCap Bench
✗	✗	59.10	76.50	78.16	1719.5	70.88	83.91	−88.67	58.20	53.31	33.18
✓	✗	57.60 0.5040 (−49.6%)	77.39 0.3554 (−64.5%)	77.76 0.7330 (−26.7%)	1624.6 0.3383 (−66.2%)	68.30 0.3759 (−62.4%)	83.50 0.6743 (−32.6%)	−86.83 0.6833 (−31.7%)	57.90 0.7002 (−30.0%)	52.53 0.3203 (−68.0%)	33.02 0.6840 (−31.6%)
✓	✓	56.00 0.3252 (−35.5%)	78.83 0.2428 (−31.7%)	78.36 0.5759 (−21.4%)	1655.1 0.2307 (−31.8%)	69.93 0.2701 (−28.1%)	82.31 0.4517 (−33.0%)	−85.67 0.3885 (−43.1%)	57.40 0.4543 (−35.1%)	53.24 0.2045 (−36.2%)	34.43 0.3909 (−42.9%)

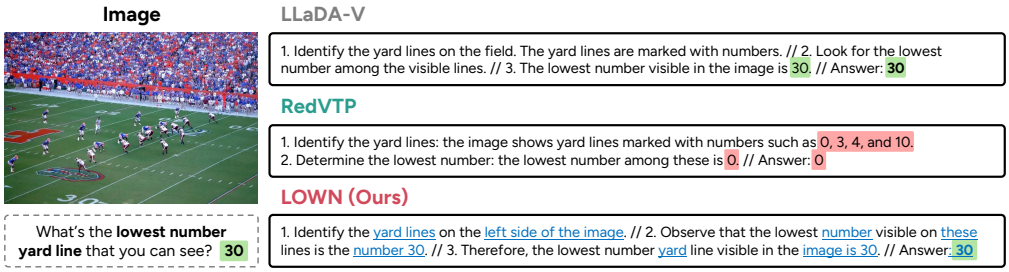


Figure 6: **Qualitative comparison with visual token pruning (RedVTP).** In LOWN's output, blue underlines mark tokens unmasked on FULL forward steps, while the remaining tokens are unmasked on TEXT-ONLY forward steps.

Visual information preservation. Visual token pruning permanently removes part of the visual representation before decoding, whereas LOWN skips image access only at selected steps and can consult the complete image later. The distinction is most apparent on tasks that require reading or composing fine-grained visual evidence, where pruning incurs its largest quality losses (Figure 5 and Table 13). In Figure 6, LLaDA-V and LOWN use the relevant visual evidence and answer correctly, whereas RedVTP prunes that evidence and answers fluently but incorrectly. Here LOWN concentrates FULL-forward steps on visually grounded content, while surrounding language is handled by TEXT-ONLY forwards.

Certificate reliability. On the baseline decoding trajectories of Section 3.2, we compare each certified token with a fresh FULL prediction. With a one-step-old anchor, the disagreement rate is 4.1–11.0%, and it generally increases with anchor age, reaching 25.3% on DocVQA when the anchor is eight steps old. On LOWN's own trajectories, the same rate is 2.2–4.7% for the tokens it unmasks on TEXT-ONLY steps. In practice the anchor rarely gets old, since every FULL forward replaces it and 92–97% of these tokens use an anchor from the preceding step, none older than five (Table 16). An oracle schedule that uses the image only on the steps where it changes an unmasked decision costs 0.33× the baseline against LOWN's measured 0.35×, and, since it still runs one forward per baseline step, it is not a lower bound. Appendix F.2 gives per-benchmark results.

6 Conclusion

DVLMs need not process visual tokens at every denoising step. Along LLaDA-V's decoding trajectories, most unmasked decisions agree between FULL and TEXT-ONLY forwards, but the steps requiring visual input vary across tasks, motivating adaptive rather than fixed image access. LOWN turns this agreement into a training- and threshold-free rule for deciding when to run a FULL forward and how many tokens to unmask. Across ten benchmarks, it retains 99.6% of LLaDA-V's score at 0.35× the decoding FLOPs. The same rule transfers unchanged to LaViDa and extends to VidLaDA under a controlled video protocol, while also composing orthogonally with visual token pruning and running under key-value caching. These results establish image-access scheduling as a broadly applicable axis for efficient DVLM decoding.

Limitations. LOWN reuses anchor predictions that agree with the TEXT-ONLY forward, so a reused token can differ from a fresh FULL prediction, more often as the anchor ages (Appendix F.2). It also reduces cost less on chart and document reading, where visual dependence persists and more steps need a FULL forward (Appendix E.5). The same agreement decides how many FULL forwards a response needs, so its cost cannot be set before decoding. Combining LOWN with visual token pruning, however, partly addresses this, as its retention ratio gives some control over cost.

AI USE STATEMENT

Generative AI tools from Anthropic and OpenAI assisted with reviewing the method implementation and experimental code for potential errors, preparing and revising figures, locating and verifying a limited set of references, and restructuring and cross-checking the manuscript. They were not used to formulate the research hypothesis, design the method or experiments, interpret the results, or select the qualitative examples reported in the paper. The authors independently validated the implementation and all AI-assisted code, checked the reported results and citations against experimental records and primary sources, and made all final research decisions. Separately, several benchmarks use LLMs as automatic judges under their standard evaluation protocols, as documented in Appendix C. The authors take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the inference efficiency of existing diffusion vision-language models on public benchmarks. It involves no human subjects and no new data collection. LOWN changes neither model weights nor training, so the models keep whatever biases and failure modes they already have, and our measurements show that LOWN largely preserves their benchmark scores rather than correcting them. Reducing decoding compute lowers the energy cost of running these models.

REPRODUCIBILITY STATEMENT

All models and benchmarks we use are publicly available. Appendix B.1 gives the complete decoding procedure, including the repetition guard, and Appendix B gives the analytical FLOPs formula for LLaDA-V. Appendix C lists the response lengths, step counts, scorers and judge versions, and the selected settings of every competitor. Appendix D describes the timing protocol, and Appendix A gives the measurement protocol behind Section 3.2. The supplementary material contains our anonymized code with a README. The code implements LOWN (Algorithm 1) on LLaDA-V, the FLOPs count of Appendix B and the measurement of Section 3.2. It runs LLaDA-V or LOWN on the ten benchmarks with the settings of Appendix C, and it includes the per-sample outputs and FLOPs of both methods with a script that recomputes their rows in the upper block of Table 1.

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*, 2021.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *The Eleventh International Conference on Learning Representations*, 2023.

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*, 2026.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, 2024.

Kanzhi Cheng, Wenpo Song, Jiaxin Fan, Zheng Ma, Qiushi Sun, Fangzhi Xu, Chenyang Yan, Nuo Chen, Jianbing Zhang, and Jiajun Chen. CapArena: Benchmarking and analyzing detailed image captioning in the LLM era. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

DiffusionGemma Team. DiffusionGemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

Mark Endo, Xiaohan Wang, and Serena Yeung-Levy. Feather the throttle: Revisiting visual token pruning for vision-language model acceleration. In *IEEE/CVF International Conference on Computer Vision*, 2025.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. MME: A comprehensive evaluation benchmark for multimodal large language models. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025.

Zhihao He, Tieyuan Chen, Kangyu Wang, Ziran Qin, Yang Shao, Chaofan Gan, Shijie Li, Zuxuan Wu, and Weiyao Lin. VidLaDA: Bidirectional diffusion large language models for efficient video understanding. *arXiv preprint arXiv:2601.17868*, 2026.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: KL-guided fast inference in masked diffusion models. In *Advances in Neural Information Processing Systems*, 2025.

Youngeun Kim, Youjia Zhang, Huiling Liu, Aecheon Jung, Sunwoo Lee, and Sungeun Hong. ZOO-Prune: Training-free token pruning via zeroth-order gradient estimation in vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Insu Lee, Wooje Park, Wonseok Shin, Jinwoo Son, and Byonghyo Shim. Visual information-guided parallel decoding for diffusion multimodal large language models. *arXiv preprint arXiv:2608.26580*, 2026.

Duo Li, Zuhao Yang, Xiaoqin Zhang, Ling Shao, and Shijian Lu. A comprehensive study on visual token redundancy for discrete diffusion-based multimodal large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, 2026a.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi, and Shiwei Liu. Diffusion language model knows the answer before it decodes. In *International Conference on Learning Representations*, 2026b.

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*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, 2024.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dLLM-Cache: Accelerating diffusion large language models with adaptive caching. In *International Conference on Machine Learning*, 2026.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 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*, 2025.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. EgoSchema: A diagnostic benchmark for very long-form video language understanding. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46212–46244, 2023.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

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*, 2026.

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*, 2025.

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*, 2024.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. In *IEEE/CVF International Conference on Computer Vision*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024.

Chengcheng Wang, Tingzhang Luo, Wenhao Li, Jianyuan Guo, and Chang Xu. TACG: Trajectory-aware commit gating for diffusion language model decoding. *arXiv preprint arXiv:2607.03236*, 2026.

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*, 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*, 2026a.

Yifan Wu, Yufeng Zhang, and Kenli Li. CORA-Diff: Confidence-oriented residual acceptance for efficient diffusion language model inference. *arXiv preprint arXiv:2608.11235*, 2026b.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Conical visual concentration for efficient large vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jingqi Xu, Jingxi Lu, Chenghao Li, Sreetama Sarkar, Souvik Kundu, and Peter A. Beerel. Red-VTP: Training-free acceleration of diffusion vision-language models inference via masked token-guided visual token pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, 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*, 2025a.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is better but not necessary in vision language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b.

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*, 2025a.

Qinghao Ye, Xianhan Zeng, Fu Li, Chunyuan Li, and Haoqi Fan. Painting with words: Elevating detailed image captioning with benchmark and alignment learning. In *International Conference on Learning Representations*, 2025b.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

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

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *International Conference on Machine Learning*, 2025.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. MLVU: Benchmarking multi-task long video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13691–13701, 2025.

A Measurement Protocol and Additional Observations

A.1 Measurement Protocol

We decode each benchmark with LLaDA-V's released procedure, greedy at temperature zero with zero classifier-free guidance and low-confidence remasking, using the response lengths and step counts of Appendix C. At every step we run the TEXT-ONLY forward alongside the FULL forward on the same partially masked response, with the visual tokens removed and the position IDs of all other tokens kept. Only the FULL forward decides which positions are unmasked, and the TEXT-ONLY outputs never enter the decoding state, so the trajectory is the baseline's by construction. At every masked position and step we log the top-1 token and its softmax confidence from both forwards, together with the positions the FULL forward unmasks.

A.2 Per-Benchmark Observations

Table 6 lists, for each benchmark, the quantities that Section 3.2 and Figure 1b summarize. The averages in the main text are unweighted means over the ten benchmarks. The FULL and TEXT-ONLY forwards agree on 71.8–92.1% of unmasked decisions, and 74.0–90.2% of final tokens were already the top-1 prediction before being unmasked. The latter is lowest on the three open-ended benchmarks, LLaVA-Bench, CapArena, and DeCapBench. Figure 7 shows the disagreement maps of Figure 2a for all ten benchmarks.

Table 6: Per-benchmark observations on LLaDA-V. Visual tokens is their mean percentage of the FULL input. Unchanged decisions is the share of unmasked positions at which the FULL and TEXT-ONLY top-1 predictions agree. Image-dependent steps is the share of steps on which the image changed at least one unmasked decision. Already top-1 is the share of final tokens that were the top-1 prediction at their position before the step that unmasked them. Brackets give 95% confidence intervals, and the last row is the unweighted mean.

Benchmark	Visual tokens (%)	Unchanged decisions	Image-dependent steps	Already top-1
MathVista	89.0	89.0 [88.6, 89.3]	17.9 [17.2, 18.5]	90.2 [89.9, 90.6]
ScienceQA	91.6	91.8 [91.0, 92.5]	8.2 [7.5, 9.0]	88.2 [87.2, 89.0]
ChartQA	96.1	75.5 [74.2, 76.7]	24.5 [23.3, 25.8]	88.4 [87.4, 89.3]
MME	97.0	89.4 [88.5, 90.3]	10.6 [9.7, 11.5]	89.6 [88.7, 90.4]
MMBench	91.3	92.1 [91.3, 92.8]	7.9 [7.2, 8.7]	88.6 [87.6, 89.5]
DocVQA	96.6	71.8 [70.9, 72.7]	43.8 [42.4, 45.2]	86.4 [85.7, 87.1]
CapArena	94.8	79.6 [79.1, 80.0]	32.7 [32.0, 33.5]	76.9 [76.5, 77.4]
LLaVA-Bench	96.2	76.3 [74.9, 77.6]	37.8 [35.6, 40.0]	74.0 [72.6, 75.3]
GQA	96.5	89.1 [88.2, 90.0]	10.9 [10.0, 11.8]	89.4 [88.5, 90.2]
DeCapBench	94.9	79.5 [79.1, 79.9]	33.4 [32.7, 34.1]	78.0 [77.6, 78.5]
Mean	94.4	83.4	22.8	85.0

A.3 Measurement on LaViDa

The measurement of Section 3.2 was repeated on LaViDa over the full evaluation sets of all ten benchmarks, with the response lengths and step counts of the LLaDA-V protocol so that the two models are measured on the same schedules. Table 7 lists the per-benchmark values. Each per-benchmark observation holds. On the eight question-answering benchmarks each measure shows less image dependence than on LLaDA-V, apart from already top-1 on MathVista, which is level, and on the two captioning benchmarks the two models are close. The image changes an unmasked decision on fewer steps, 1.3–30.6% against 7.9–43.8% on LLaDA-V, the two forwards agree on 81.6–98.7% of unmasked decisions, and 78.6–93.7% of final tokens were the top-1 prediction before they were unmasked.

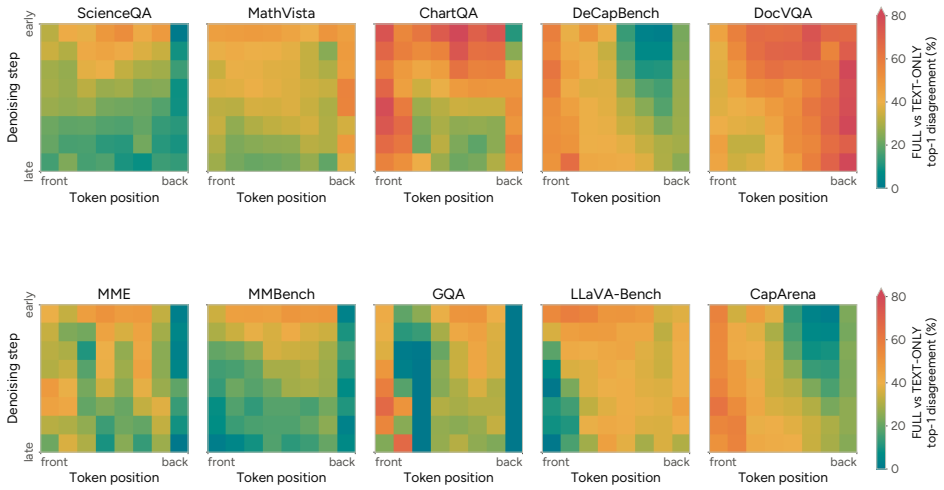


Figure 7: **Where and when the image changes a prediction, all ten benchmarks.** Share of masked positions at which the FULL and TEXT-ONLY top-1 predictions disagree, over token position (front to back) and denoising step (early to late), as in Figure 2a.

Table 7: Per-benchmark observations on LaViDa, in the format of Table 6, with the response lengths and step counts of the LLaDA-V measurement and over the full evaluation sets. Brackets give 95% confidence intervals, and the last row is the unweighted mean.

Benchmark	Visual tokens (%)	Unchanged decisions	Image-dependent steps	Already top-1
MathVista	71.1	94.0 [93.8, 94.1]	10.4 [10.1, 10.7]	89.8 [89.6, 90.0]
ScienceQA	76.4	98.7 [98.6, 98.8]	1.3 [1.2, 1.4]	93.7 [93.5, 94.0]
ChartQA	88.8	83.6 [83.2, 84.0]	16.4 [16.0, 16.8]	92.6 [92.4, 92.9]
MME	87.9	96.1 [95.9, 96.3]	3.9 [3.7, 4.1]	93.7 [93.5, 94.0]
MMBench	78.4	96.8 [96.6, 96.9]	3.2 [3.1, 3.4]	93.7 [93.5, 93.9]
DocVQA	88.3	89.4 [89.3, 89.6]	15.2 [14.9, 15.4]	91.6 [91.4, 91.7]
CapArena	81.1	82.0 [81.7, 82.3]	29.8 [29.3, 30.2]	78.6 [78.3, 78.9]
LLaVA-Bench	86.3	91.3 [90.4, 92.1]	13.6 [12.2, 15.1]	88.1 [87.1, 89.0]
GQA	88.8	95.8 [95.7, 95.9]	4.2 [4.1, 4.3]	93.1 [92.9, 93.2]
DeCapBench	80.7	81.6 [81.2, 81.9]	30.6 [30.0, 31.2]	78.6 [78.2, 79.0]
Mean	82.8	90.9	12.9	89.3

B Method Details and Cost Accounting

This appendix gives the complete decoding procedure, restates the cost model of Section 4.3 and derives its forward cost $\mathcal{C}(n)$, and writes visual token pruning in the same form as Equation 3.

B.1 Detailed Decoding Algorithm

Each response starts fully masked, with its length and baseline unmasking schedule set by the benchmark protocol. The first iteration runs TEXT-ONLY followed by FULL on the same response, establishing the anchor and unmasking k positions. Confidence is the softmax probability of the top-1 token. The repetition guard ends the certified prefix before a candidate whose token ID matches either immediate neighbor that is already unmasked or was accepted earlier in the same scan. The guard is needed because the positions accepted on one TEXT-ONLY step are predicted independently of one another, so two adjacent positions can receive the same token, and without the guard such pairs would be unmasked together and appear as repeated words in the response. Algorithm 1 gives the full procedure.

Algorithm 1 LOWN decoding

Require: visual tokens v, prompt tokens c, tokens unmasked per step k
Ensure: response x

$x \leftarrow$ fully masked response; $\hat{y}^{\text{text}} \leftarrow \arg\max p_\theta(\cdot \mid c, x)$ ▷ draft for the first forward
while x has masked positions **do**
  $\bar{y} \leftarrow \arg\max p_\theta(\cdot \mid v, c, x)$, $\bar{s} \leftarrow p_\theta(\bar{y} \mid v, c, x)$ ▷ FULL forward: update anchor
  $\mathcal{U}^{\text{full}} \leftarrow$ top-k masked positions by $\bar{s}$, those with $\hat{y}_i^{\text{text}} \neq \bar{y}_i$ first
  $x_i \leftarrow \bar{y}_i$ for $i \in \mathcal{U}^{\text{full}}$
  repeat
    $\hat{y}^{\text{text}} \leftarrow \arg\max p_\theta(\cdot \mid c, x)$ ▷ TEXT-ONLY forward
    $M \leftarrow$ masked positions in decreasing $\bar{s}$
    $\mathcal{U}^{\text{text}} \leftarrow$ longest prefix of M with $\hat{y}_i^{\text{text}} = \bar{y}_i$ ▷ certificate
    if $|\mathcal{U}^{\text{text}}| \geq k$ **then** $x_i \leftarrow \bar{y}_i$ for $i \in \mathcal{U}^{\text{text}}$ ▷ anchor kept
    if x has no masked positions **then break**
  until $|\mathcal{U}^{\text{text}}| < k$ ▷ a certificate fails
end while
return x

B.2 Cost of One Forward

Section 4.3 writes the cost of LOWN relative to the baseline as $R = f + \sigma\rho$ (Equation 3). The baseline runs T FULL forwards over $N_F = |v| + |c| + |x_t|$ tokens. LOWN runs n_T TEXT-ONLY forwards over $N_T = |c| + |x_t|$ tokens and adds a FULL forward on n_F of those steps, so $f = n_F/T$ is the frequency of FULL forwards, $\sigma = n_T/T$ the relative number of steps, and $\rho = \mathcal{C}(N_T)/\mathcal{C}(N_F)$ the cost of a TEXT-ONLY forward relative to a FULL one, where $\mathcal{C}(n)$ is the FLOPs of one forward over n tokens. A FULL step is charged both forwards. This section gives $\mathcal{C}(n)$.

We count matrix multiplications only, with one multiply-add as two FLOPs. For a model with L layers, hidden width d, FFN width m, and vocabulary $\mathcal{V}$ (for LLaDA-V, L=32, d=4096, m=12288, $|\mathcal{V}|$=126,349), one layer over n tokens costs

$$\mathcal{C}_{\text{layer}}(n) = \underbrace{8nd^2 + 4n^2d}_{\text{attention}} + \underbrace{6ndm}_{\text{FFN}}, \qquad \mathcal{C}(n) = L \cdot \mathcal{C}_{\text{layer}}(n) + 2nd|\mathcal{V}|, \tag{4}$$

where $8nd^2$ is the Q/K/V/O projections, $4n^2d$ is $QK^\top$ and AV and is quadratic in length, $6ndm$ is the three matrices of the gated FFN, and the last term of $\mathcal{C}(n)$ is the LM head. The quadratic term is kept because it is what makes ρ smaller than the length ratio N_T/N_F, since removing v shrinks n by roughly 9–33$\times$ and the quadratic term by roughly the square of that. The count excludes the vision encoder, which runs once per sample and would add 0.16–1.89% of the baseline decoding cost on the ten benchmarks, the same for every method on a benchmark, and non-matmul operations such as softmax and normalization. Table 8 reports the mean ρ per benchmark, and Appendix D reports measured generation time.

B.3 Other Methods in the Same Form

Relative to the unpruned baseline, visual token pruning costs π, LOWN costs $f + \sigma\rho$, and the two together cost $\pi f + \sigma\rho$. Pruning fixes its budget π before decoding starts, whereas LOWN decides f and σ during decoding, per input and per step. Key-value caching instead lowers the cost of every forward that reuses the cache, and LOWN's acceptance lowers the number of forwards, which is the saving measured in Table 2.

Table 8: Mean TEXT-ONLY/FULL forward-cost ratio on LLaDA-V. For each sample, we compute $\rho_i = \mathcal{C}(p_i + o_i)/\mathcal{C}(p_i + v_i + o_i)$ using its prompt length p_i, visual-token count v_i, and configured response length o_i. Each row averages these ratios over the benchmark's samples, and $\mathcal{C}$ is the analytical forward cost in Equation 4.

Benchmark	Samples	Mean ρ (%)
MathVista	1,000	10.07
ScienceQA	2,017	7.92
ChartQA	2,500	2.54
MME	2,374	3.15
MMBench	4,329	7.04
DocVQA	5,349	3.02
CapArena	600	4.63
LLaVA-Bench	60	3.35
GQA	12,578	3.20
DeCapBench	400	4.49

C Experimental Setup

C.1 Benchmarks and Settings

Table 8 lists the evaluated sample counts for LLaDA-V. We use greedy decoding with temperature zero and classifier-free guidance zero, with low-confidence remasking. The configured response length and baseline denoising steps are 100/50 for MathVista, 32/16 for DocVQA, 64/32 for LLaVA-Bench, 128/64 for CapArena and DeCapBench, and 16/16 for ScienceQA, ChartQA, MME, MMBench, and GQA. ChartQA generation ends at the first newline. The baseline unmasks k tokens per step, the response length divided by the number of steps, so $k = 2$ on MathVista, DocVQA, LLaVA-Bench, CapArena, and DeCapBench and $k = 1$ on the others. MathVista uses the model's reasoning prompt, and the other tasks use the non-reasoning setting. These response budgets are held fixed within each comparison. For standalone LLaDA-V, we use the authors' RedVTP code and adapt the released ZOO-Prune and KLASS implementations. The released ZOO-Prune code targets autoregressive LLaVA models. We keep its token scoring and selection, select the visual tokens once before decoding, and use the retained tokens at every denoising step. Reported FLOPs use the tokens actually retained and include the scoring cost. Composition comparisons target the corresponding LOWN budget and include existing off-grid settings. Table 9 specifies the retention and KLASS thresholds selected for Table 1. The Avg. column in Table 1 is the unweighted mean of $100\times$score/baseline score, except that CapArena uses $100+$score$-$baseline score.

C.2 Judges

MME reports perception plus cognition for LLaDA-V and perception alone for LaViDa, while CapArena reports a win–loss score. We score ScienceQA, ChartQA, MME, DocVQA, and GQA with task-specific rules. For LLaDA-V ScienceQA, we extract answers with the deterministic multiple-choice parser from lmms-eval v0.7.1. MathVista extracts final answers with GPT-4o-2024-08-06 before scoring, and CapArena is judged with the same model. MMBench uses rule matching followed by GPT-3.5-turbo-0125 when needed, scoring unmatched answers as incorrect, and LLaVA-Bench is judged with GPT-4o-2024-11-20. DeCapBench uses Qwen3.5-35B-A3B for its fact-based evaluation. All methods within each reported comparison use the same scorer.

C.3 Baseline Settings

Each competitor exposes a budget hyperparameter, the retention ratio r for RedVTP and ZOO-Prune and the confidence and KL thresholds (τ, κ) for KLASS. We run each on a fixed grid whose ends keep the method within its working range, a 10% grid of r for RedVTP and ZOO-Prune on LLaDA-V together with existing off-grid settings, and a grid of threshold pairs for KLASS. Because the grid is discrete, a competitor's FLOPs cannot be set equal to LOWN's. For every benchmark we therefore select the cheapest grid setting whose FLOPs are at least those of the LOWN row it is compared with, and scores play no part in the selection. Every competitor cell in Tables 1 and 13

consequently spends at least as much compute as LOWN, so a competitor that scores below LOWN does so at equal or higher cost. Table 9 lists the selected settings on LLaDA-V, and Table 13 shows the selected retention under each LaViDa cell.

Figure 1c matches scores instead of FLOPs. For each benchmark the target is LOWN's score, and each pruning method is drawn at its cheapest measured setting, up to 75% retention, whose score is at least the target. When no setting up to 75% reaches the target, the 75% setting is drawn, so its bar is an upper bound on the speedup the method would have at LOWN's score. Only measured settings are used, without interpolation.

Table 9: Settings behind the competitor cells of Table 1 (FLOPs matched) and Table 10 (time matched). For RedVTP, r is the visual-token retention percentage. For ZOO-Prune, the released code sets the budget to $r\%$ of a fixed 2,880 tokens rather than of each image's visual tokens, so the fraction actually retained varies with the image. For KLASS, (τ, κ) denotes its confidence and KL thresholds. The second and fourth blocks use the budget of LOWN composed with RedVTP at $r = 50\%$.

	MathVista	ScienceQA	ChartQA	MME	MMBench	DocVQA	CapArena	LLaVA-B.	GQA	DeCapB.
FLOPs matched to LOWN										
RedVTP r	30	20	60	20	20	50	40	50	20	40
ZOO-Prune r	30	20	80	30	20	60	50	60	20	50
KLASS (τ, κ)	(0.2, 0.2)	(0.2, 1)	(0.2, 1)	(0.2, 0.3)	(0.2, 0.4)	(0.2, 0.1)	(0.2, 0.5)	(0.2, 1)	(0.2, 1)	(0.2, 0.4)
FLOPs matched to LOWN + RedVTP ($r = 50\%$)										
RedVTP r	20	10	30	10	20	25	20	25	10	20
ZOO-Prune r	11.12	11.12	50	20	11.12	33.34	30	33.34	11.12	30
KLASS + RedVTP (τ, κ)	(0.2, 0.1)	(0.2, 0.4)	(0.1, 0.2)	(0.2, 0.3)	(0.2, 0.4)	(0.2, 0.1)	(0.2, 0.5)	(0.2, 1)	(0.2, 1)	(0.2, 0.4)
Time matched to LOWN										
RedVTP r	30	10	50	10	10	50	50	60	10	50
ZOO-Prune r	30	11.12	75	30	22.23	60	60	70	22.23	60
KLASS (τ, κ)	(0.3, 0.1)	(0.2, 0.4)	(0.2, 0.5)	(0.2, 0.3)	(0.2, 0.1)	(0.3, 0.1)	(0.2, 0.5)	(0.2, 0.5)	(0.2, 0.1)	(0.2, 0.4)
Time matched to LOWN + RedVTP ($r = 50\%$)										
RedVTP r	10	10	40	10	10	30	25	40	10	25
ZOO-Prune r	11.12	11.12	70	30	22.23	50	33.34	40	22.23	33.34
KLASS + RedVTP (τ, κ)	(0.4, 0.1)	(0.5, 0.1)	(0.2, 0.1)	(0.3, 0.1)	(0.5, 0.1)	(0.5, 0.1)	(0.2, 0.1)	(0.2, 0.1)	(0.3, 0.1)	(0.2, 0.1)

C.4 Video Evaluation

We evaluate the 500-question EgoSchema subset and the 502-question MLVU test set. We use VidLaDA-8B with 32 frames per video, sampled at one frame per second and uniformly subsampled when needed. Each frame yields 210 visual tokens after spatial pooling. We adopt a controlled decoding protocol with a 128-token response, 128 baseline diffusion steps, and block length 32, following the efficiency-ablation setup of He et al. (2026). Decoding is greedy, with zero classifier-free guidance and low-confidence remasking. The baseline's parallel decoding is disabled, and LOWN uses adaptive acceptance within each block. We ask the model to reason inside `<think>` and `</think>` before answering with an option letter. This protocol differs from VidLaDA's released 64-frame, two-token evaluation.

The two methods are evaluated on identical questions and frames. We score only the answer span after the closing tag, using the released task parsers. EgoSchema reports accuracy. MLVU reports the mean accuracy over its nine task types (M-Avg). For each benchmark, the FLOPs ratio divides LOWN's summed decoding FLOPs by the baseline's sum over the same questions. It includes the decoder and LM head and excludes the vision encoder.

D Wall-Clock Measurements

D.1 Generation Time

All LLaDA-V timing experiments were run on the same server with NVIDIA H200 GPUs, with one experiment per GPU at a time. We measure per-example generation time at batch size 1 with a host clock around the model's generation call. CUDA is synchronized before the timer starts and after generation returns. The interval includes decoding, multimodal preparation, and vision encoding performed within the call. Model loading and evaluation scoring are excluded. Each task mean

averages all per-example times from one recorded execution of the full evaluation set. FLOPs count the decoder, LM head, and method-specific matrix multiplications, so their scope differs from the timing interval. The cache comparison in Table 2 includes 1,000 MathVista examples, 60 LLaVA-Bench questions, and 400 DeCapBench examples.

D.2 Quality and Generation Time

Table 10 repeats the comparison of Table 1 with each competitor selected by measured time. For each competitor, among its measured retentions (down to the floors of Table 1) and KLASS thresholds, we take the setting with the shortest mean generation time that is at least LOWN's, so no competitor is given less time than LOWN. When even the fastest setting is slower than LOWN, that setting is used. Accuracy plays no part in the selection. Cells within a few percent of LOWN's time should be read as equal in cost. With every competitor given at least LOWN's generation time on every benchmark, LOWN keeps the highest average in both blocks, 99.6% and 98.3% of LLaDA-V against at most 94.6% and 92.8% for the competitors, and scores highest on seven and eight of the ten benchmarks, respectively.

Table 10: **Performance at matched generation time on LLaDA-V**. The upper block compares standalone methods at LOWN's generation time, and the lower block at that of LOWN with RedVTP (r=50%). Competitors run on a discrete hyperparameter grid, so each cell uses the setting with the shortest measured time at or above LOWN's (Table 9), and LOWN gives the highest normalized average in both blocks. Cells show scores with mean generation time relative to LLaDA-V below. Best scores are bold, second best underlined, and lowest times bold.

Method	Math Vista	Science QA	Chart QA	MME	MM Bench	Doc VQA	Cap Arena	LLaVA-Bench	GQA	DeCap Bench	Avg.
LLaDA-V *(Vanilla)*	59.10	76.50	78.16	1719.49	70.88	83.91	−88.67	58.20	53.31	33.18	100.0%
Time matched to LOWN											
RedVTP	51.70 (×0.394)	65.49 (×0.374)	72.20 (×0.674)	1275.08 (×0.300)	40.12 (×0.366)	**83.39** (×0.566)	−88.50 (×0.507)	**57.60** (×0.612)	48.05 (×0.318)	33.88 (×0.504)	88.7% (×0.461)
ZOO-Prune	53.70 (×0.417)	72.48 (×0.331)	76.00 (×0.690)	1582.41 (×0.352)	68.73 (×0.393)	78.73 (×0.548)	−94.67 (×0.490)	56.70 (×0.587)	50.50 (×0.301)	31.15 (×0.488)	94.6% (×0.460)
KLASS	**56.90** (×0.493)	76.05 (×0.337)	73.56 (×0.676)	1641.99 (×0.309)	68.56 (×0.387)	76.67 (×0.560)	−96.83 (×0.450)	42.20 (×0.573)	49.55 (×0.284)	30.18 (×0.479)	92.2% (×0.455)
LOWN (Ours)	56.00 **(×0.380)**	**78.83** **(×0.331)**	**78.36** **(×0.666)**	**1655.08** **(×0.295)**	**69.93** **(×0.362)**	82.31 **(×0.537)**	**−85.67** **(×0.441)**	57.40 **(×0.532)**	**53.24** **(×0.268)**	**34.43** **(×0.446)**	**99.6%** **(×0.426)**
Time matched to LOWN (Ours) + RedVTP (r=50%)											
RedVTP	41.70 (×0.279)	65.49 (×0.374)	70.80 (×0.635)	1275.08 (×0.300)	40.12 (×0.366)	82.14 (×0.408)	−89.33 (×0.299)	**57.10** (×0.436)	48.05 (×0.318)	32.28 (×0.296)	86.0% (×0.371)
ZOO-Prune	46.10 (×0.264)	72.48 (×0.331)	**74.92** (×0.648)	1582.41 (×0.352)	68.73 (×0.393)	75.97 (×0.453)	−96.17 (×0.301)	52.90 (×0.357)	50.50 (×0.301)	29.56 (×0.299)	91.5% (×0.370)
KLASS + RedVTP *(r=50%)*	54.70 (×0.327)	74.67 (×0.370)	70.64 (×0.657)	1606.29 (×0.306)	68.38 (×0.364)	82.07 (×0.418)	−96.00 (×0.296)	47.20 (×0.371)	50.74 (×0.261)	30.13 (×0.299)	92.8% (×0.367)
LOWN (Ours) + RedVTP *(r=50%)*	**54.90** **(×0.262)**	**75.56** **(×0.319)**	74.88 **(×0.577)**	**1709.75** **(×0.276)**	**69.67** **(×0.333)**	**82.24** **(×0.398)**	**−85.17** **(×0.284)**	55.00 **(×0.354)**	**53.42** **(×0.256)**	**33.61** **(×0.278)**	**98.3%** **(×0.334)**

D.3 Cache Configuration and Composition

Fast-dLLM uses a block spanning the configured output, prefix refresh every 32 forwards, and its factor-based token selection with factor 1. The combined implementation retains these cache settings, maintains separate image-conditioned and TEXT-ONLY prefix caches, and takes the union of Fast-dLLM's selected tokens and LOWN's accepted tokens.

E Additional Results

E.1 Fixed-Retention Pruning Compositions

Table 11 measures the effect of adding LOWN to the unpruned model and to each fixed pruner, using the corresponding configuration without LOWN as the reference. LOWN retains much of its

standalone compute reduction when applied on top of pruning, with average quality close to that of each reference configuration.

Table 11: Relative gains from adding LOWN on LLaDA-V and LaViDa. Each row is measured against the listed configuration without LOWN. FLOPs ratio is the mean of the per-benchmark FLOPs ratios, and reduction is one minus this mean. Quality is the mean normalized index with the reference set to 100.

Configuration before LOWN	FLOPs ratio ↓	Reduction ↑	Quality index ↑
LLaDA-V			
LLaDA-V (unpruned)	0.353	64.7%	99.6
RedVTP (50% retention)	0.424	57.6%	100.9
ZOO-Prune (33% retention)	0.394	60.6%	98.7
LaViDa			
LaViDa (unpruned)	0.174	82.6%	99.7
RedVTP (75% retention)	0.227	77.3%	99.5
ZOO-Prune (33% retention)	0.278	72.2%	99.8

E.2 Fewer Denoising Steps

Table 12 runs LLaDA-V with $T/2$ and $T/4$ denoising steps on the CapArena, LLaVA-Bench, and DeCapBench inputs of Table 1. LOWN's FLOPs lie between the two settings, and LOWN scores above both on all three benchmarks, including $T/2$ at $0.500\times$.

Table 12: LLaDA-V with fewer denoising steps on CapArena, LLaVA-Bench, and DeCapBench. $T/2$ and $T/4$ run half and a quarter of the baseline's T steps on the inputs of Table 1. Scores use the judges of Table 1, and FLOPs are relative to LLaDA-V. DeCapBench averages the captions its official evaluator judges valid, 399, 398, 392, and 399 of 400 for the four rows.

	CapArena		LLaVA-Bench		DeCapBench	
Method	Score ↑	FLOPs ↓	Score ↑	FLOPs ↓	Score ↑	FLOPs ↓
LLaDA-V (T steps)	−88.67	1.000	58.20	1.000	33.18	1.000
LLaDA-V ($T/2$)	−92.33	0.500	45.40	0.500	32.17	0.500
LLaDA-V ($T/4$)	−99.00	0.250	26.20	0.250	25.84	0.250
LOWN (Ours)	−85.67	0.388	57.40	0.454	34.43	0.391

E.3 LaViDa

Figure 8 shows all ten benchmarks, and Table 13 gives the FLOPs-matched comparison in the format of Table 1.

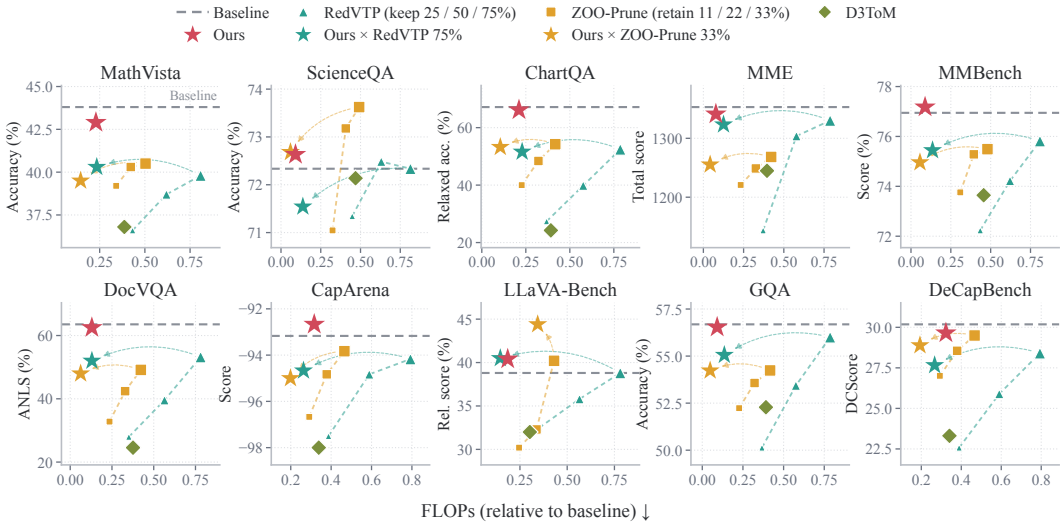


Figure 8: **Performance and efficiency on LaViDa, all ten benchmarks.** The markers follow Figure 5.

Table 13: **Performance at matched decoding FLOPs on LaViDa**, in the format of Table 1. Methods are compared at LOWN's budget. Each cell shows the benchmark score, with the selected retention r and FLOPs relative to the LaViDa baseline below. Competitors run on a discrete grid of retention ratios, so each uses its cheapest measured setting whose FLOPs are at least LOWN's (Appendix C.3). D^{3}ToM uses constant 10% retention on MathVista, LLaVA-Bench, CapArena, and DeCapBench and its adaptive schedule on the remaining tasks. MME reports the perception score. Best scores are bold, second best underlined, and lowest FLOPs bold.

Method	Math Vista	Science QA	Chart QA	MME	MM Bench	Doc VQA	Cap Arena	LLaVA-Bench	GQA	DeCap Bench	Avg.
LaViDa *(Vanilla)*	43.80	72.34	67.16	1353.03	76.95	63.54	−93.17	38.80	56.69	30.18	100.0%
FLOPs matched to LOWN											
RedVTP	36.60 r=25% (×0.432)	71.34 r=25% (×0.448)	20.40 r=10% (×0.244)	987.28 r=10% (×0.248)	72.23 r=25% (×0.435)	18.55 r=10% (×0.225)	−99.33 r=20% (×0.347)	28.60 r=10% (×0.226)	50.14 r=25% (×0.367)	22.58 r=25% (×0.389)	73.9% (×0.336)
ZOO-Prune	39.20 r=11% (×0.341)	71.05 r=11% (×0.324)	39.56 r=10% (×0.217)	1220.76 r=11% (×0.232)	73.76 r=11% (×0.309)	30.62 r=10% (×0.225)	−95.67 r=15% (×0.321)	30.20 r=11% (×0.245)	52.23 r=11% (×0.227)	28.55 r=22% (×0.379)	84.3% (×0.282)
D^{3}ToM	36.80 (×0.387)	72.14 (×0.469)	24.24 (×0.392)	1244.92 (×0.396)	73.64 (×0.456)	24.56 (×0.376)	−98.00 (×0.339)	32.00 (×0.302)	52.28 (×0.391)	23.31 (×0.341)	79.3% (×0.385)
LOWN (Ours)	**42.90** **(×0.229)**	**72.63** **(×0.091)**	**66.16** **(×0.211)**	**1341.15** **(×0.078)**	**77.18** **(×0.088)**	**62.49** **(×0.130)**	**−92.67** **(×0.316)**	**40.40** **(×0.185)**	**56.54** **(×0.091)**	**29.65** **(×0.323)**	**99.7%** **(×0.174)**

E.4 Qualitative Samples

Figures 9 and 10 extend the comparison of Figure 6 to more examples. In Figure 9, RedVTP answers the short-answer questions incorrectly and, in open-ended captioning, describes objects and attributes that do not match the image. LOWN gives the same answers as LLaDA-V and describes the same main objects. In Figure 10, RedVTP processes the image at every step, but after pruning it processes only the visual tokens it retains, which exclude the x-axis tick 4.0 that answers the question. Its response is fluent but ends with the incorrect answer 2.5. LOWN processes the image on fewer steps, but each FULL forward covers the complete image.

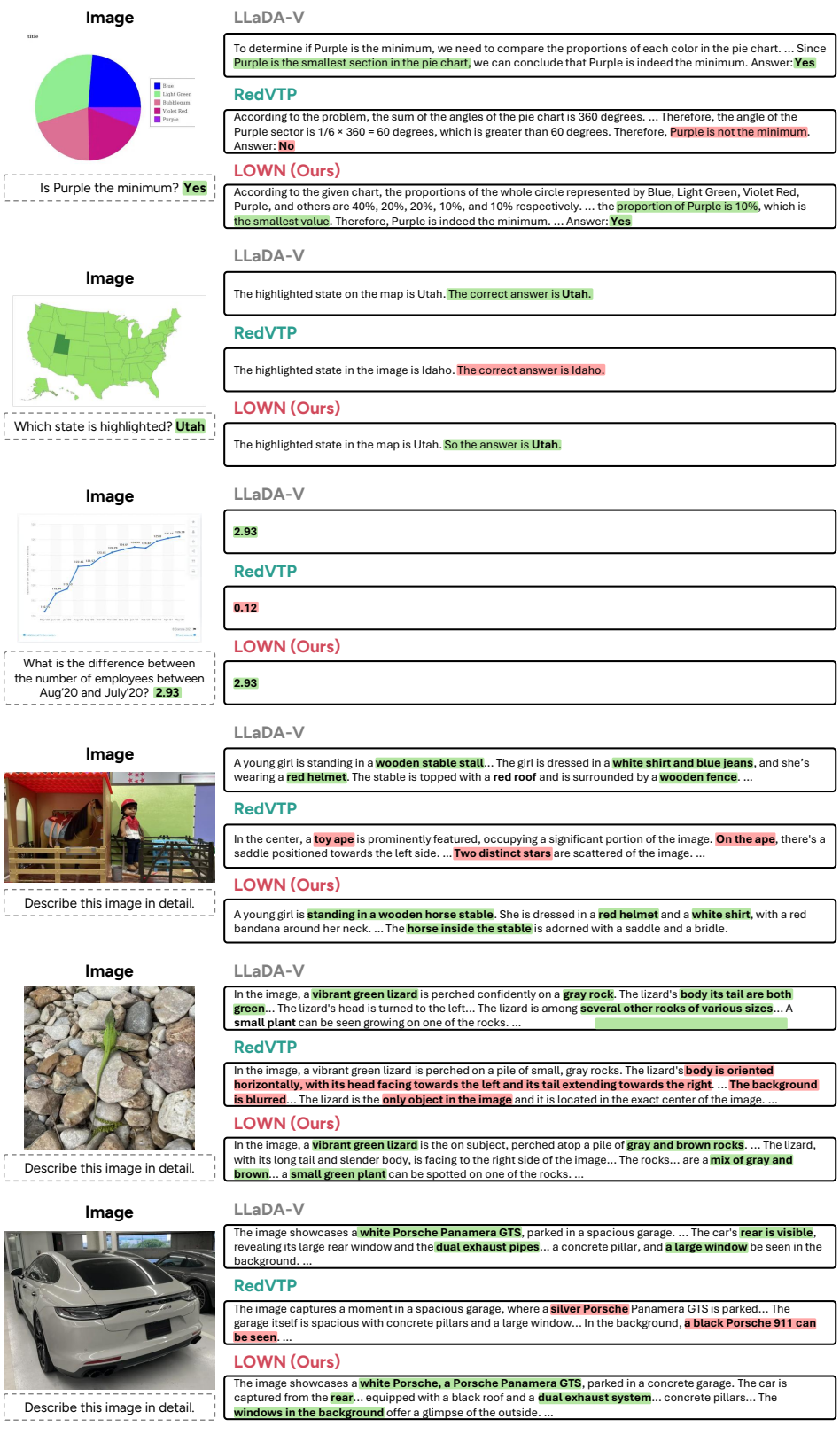


Figure 9: **Additional qualitative comparisons with visual token pruning (RedVTP).** The upper three examples are short-answer questions, and the lower three are open-ended captioning. Green marks correct answers and content consistent with the image, and red marks incorrect answers and content inconsistent with the image.

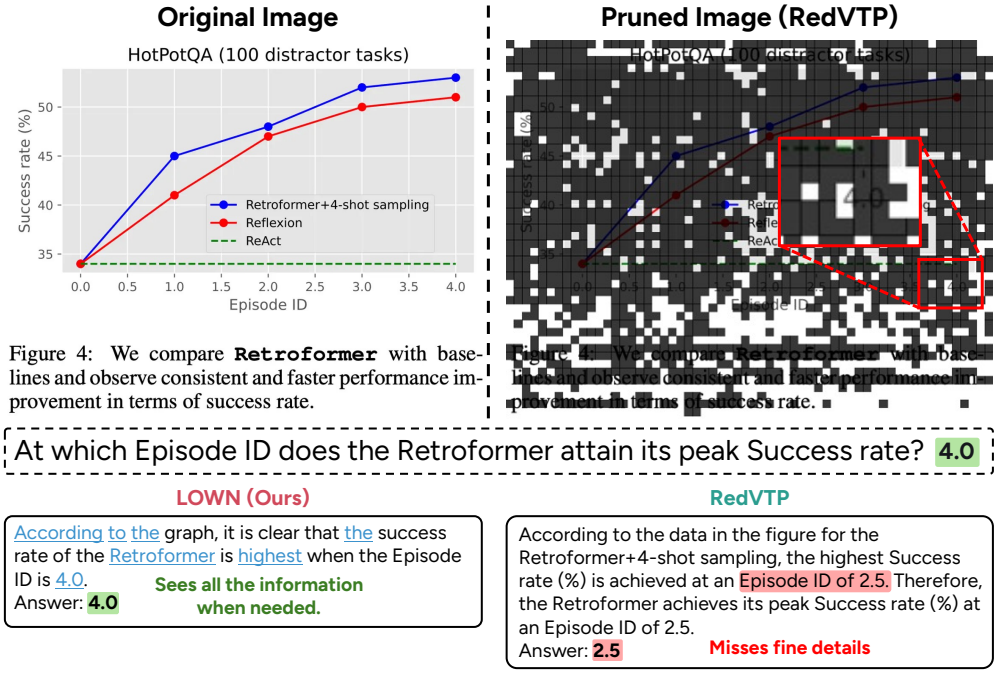


Figure 10: **Qualitative comparison with visual token pruning (RedVTP) and its pruned image.** The upper right darkens the visual tokens removed by RedVTP, and the inset enlarges the region around the x-axis tick 4.0. In LOWN's output, blue underlines mark tokens unmasked on FULL forward steps, while the remaining tokens are unmasked on TEXT-ONLY forward steps.

E.5 Limitation Example

Figure 11 shows the DocVQA input on which LOWN saves the least. The question asks for a number in a table, and LOWN runs a FULL forward on 14 of its 15 steps. LOWN answers correctly but costs 0.90× the baseline FLOPs, against 0.45× over the whole benchmark (Table 1).

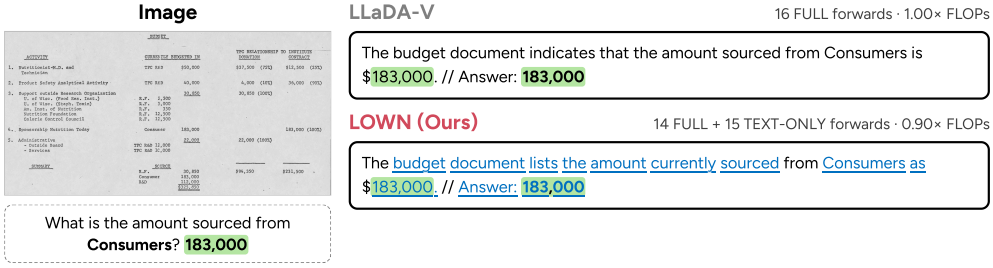


Figure 11: **A limitation example on DocVQA.** In LOWN's output, blue underlines mark tokens unmasked on FULL forward steps, while the remaining tokens are unmasked on TEXT-ONLY forward steps. FLOPs are relative to LLaDA-V on this input.

F Additional Ablations and Analysis

F.1 Order of FULL-Step Unmasking

Table 14 separates the two parts of adaptive acceptance in Table 5. Disagreement-first order removes 14–35% of the cost of visual routing alone while scoring higher on seven of the ten benchmarks, and whole-prefix acceptance removes a further 2–19%.

Table 14: Table 5 with adaptive acceptance split into disagreement-first order on FULL steps and whole-prefix acceptance on TEXT-ONLY steps (Section 4.2), on LLaDA-V. Each setting reports scores and, below, relative FLOPs with the reduction from the preceding setting in parentheses.

Visual routing	Adaptive acceptance		Math Vista	Science QA	Chart QA	MME	MM Bench	Doc VQA	Cap Arena	LLaVA-Bench	GQA	DeCap Bench
	Disagreement	Whole prefix										
✗	✗	✗	59.10	76.50	78.16	1719.5	70.88	83.91	−88.67	58.20	53.31	33.18
✓	✗	✗	57.60 0.5040 (−49.6%)	77.39 0.3554 (−64.5%)	77.76 0.7330 (−26.7%)	1624.6 0.3383 (−66.2%)	68.30 0.3759 (−62.4%)	83.50 0.6743 (−32.6%)	−86.83 0.6833 (−31.7%)	57.90 0.7002 (−30.0%)	52.53 0.3203 (−68.0%)	33.02 0.6840 (−31.6%)
✓	✓	✗	57.10 0.3910 (−22.4%)	78.19 0.3006 (−15.4%)	78.32 0.5891 (−19.6%)	1688.5 0.2755 (−18.6%)	69.85 0.3236 (−13.9%)	82.33 0.4789 (−29.0%)	−81.17 0.4471 (−34.6%)	57.50 0.5067 (−27.6%)	53.43 0.2469 (−22.9%)	34.71 0.4450 (−34.9%)
✓	✓	✓	56.00 0.3252 (−16.8%)	78.83 0.2428 (−19.2%)	78.36 0.5759 (−2.2%)	1655.1 0.2307 (−16.3%)	69.93 0.2701 (−16.5%)	82.31 0.4517 (−5.7%)	−85.67 0.3885 (−13.1%)	57.40 0.4543 (−10.3%)	53.24 0.2045 (−17.2%)	34.43 0.3909 (−12.2%)

F.2 Certificate Reliability

Coverage, $\epsilon(\Delta)$, and R_{ref} are computed on the logged baseline trajectories of Appendix A.1, and the acceptance error and anchor age on LOWN's own decoding of the same samples (Tables 15 and 16).

At anchor age one the certificate certifies 43–72% of the eligible positions, $\epsilon(\Delta)$ rises from age one to four on every benchmark, and the acceptance error stays below $\epsilon(1)$ on every benchmark. On DocVQA, LOWN costs less than the oracle schedule because adaptive acceptance ends the trajectory early.

Table 15: Certificate reliability on LLaDA-V. Eligible positions are the masked positions that were already masked Δ steps earlier. Coverage is the share of eligible positions whose TEXT-ONLY prediction agrees with a one-step-old anchor. $\epsilon(\Delta)$ is the share of agreeing positions at which a fresh FULL forward predicts a different token when the anchor is Δ steps old. $R_{\text{ref}} = f^*+(1-f^*)\rho+1/T$ is the cost of the oracle schedule of Section 5.3, which runs a FULL forward only on the fraction f^* of steps where the image changes an unmasked decision and pays ρ elsewhere, plus one initial FULL forward. For ChartQA, whose baseline stops at the end-of-sequence token after 7.0 of its 16 steps on average, f^* and T are taken over the steps the baseline actually runs ($f^* = 32.3\%$). Acceptance error is the share of the tokens LOWN unmasks on TEXT-ONLY steps that a fresh FULL forward on the state they were accepted from would predict differently, and R_{LOWN} is LOWN's measured cost. Costs are relative to the baseline and rates are in percent.

Benchmark	Coverage	$\epsilon(1)$	$\epsilon(2)$	$\epsilon(4)$	$\epsilon(8)$	R_{ref}	Accept. error	R_{LOWN}
MathVista	58.6	7.3	9.5	12.2	15.4	0.284	2.2	0.325
ScienceQA	63.5	6.4	8.3	9.2	8.8	0.218	4.7	0.243
ChartQA	47.0	10.6	14.0	16.6	15.2	0.490	4.2	0.576
MME	63.8	5.9	7.6	10.0	13.6	0.193	3.9	0.231
MMBench	65.3	6.3	7.3	7.5	6.9	0.217	3.9	0.270
DocVQA	43.4	11.0	15.5	20.2	25.3	0.517	3.0	0.452
CapArena	72.1	5.2	6.8	8.7	10.8	0.374	3.5	0.388
LLaVA-Bench	54.0	9.5	12.6	16.1	19.5	0.430	4.2	0.454
GQA	64.7	4.1	5.4	7.0	11.1	0.200	2.3	0.205
DeCapBench	70.6	5.5	7.3	9.2	11.6	0.380	3.4	0.391
Mean						0.330		0.353

Table 16: Anchor age of the tokens LOWN accepts on TEXT-ONLY steps, in LOWN's own runs on LLaDA-V. Age is the number of LOWN steps since the FULL forward that produced the anchor. Shares are in percent, and the last row pools the accepted tokens of all benchmarks.

Benchmark	Age 1	Age ≤ 2	Max age
MathVista	93.8	99.5	5
ScienceQA	92.9	99.4	3
ChartQA	94.9	99.9	3
MME	94.2	99.5	4
MMBench	91.8	99.4	3
DocVQA	96.7	99.6	4
CapArena	95.4	99.8	3
LLaVA-Bench	93.9	99.7	4
GQA	94.8	99.5	4
DeCapBench	95.3	99.9	3
All	94.8	99.7	5