PACE: PRUNE AGGRESSIVELY AND COMMIT EARLY FOR DIFFUSION VLM INFERENCE ACCELERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion vision-language models (DVLMs) have emerged as an alternative to autoregressive VLMs for their capacity for parallel decoding. In practice, however, each denoising step runs full bidirectional attention over the entire sequence, and the resulting inference cost scales as the product of the step count and the per-step cost of a full-sequence pass. Since visual tokens make up most of the sequence, compressing them has become an active direction. We show that the step count is a design factor as important as compression for DVLM inference acceleration, and that the two are not separable. Compression is applied after the first denoising step, which commits part of the answer on the full visual context, and the step count determines the size of that part. Under the default schedule, the core content of the answer is determined on the pruned context, where aggressive compression loses its accuracy. With fewer steps, more of the answer is committed before pruning and less of it is exposed to the loss from compression, so the penalty shrinks. We formalize this regularity as a monotone comparative statics result. Based on these observations, we propose PACE, a training-free inference acceleration framework for DVLMs that scores visual tokens from the first forward pass and selects the step count jointly with the retention ratio under a task-level rule. Reducing the step count or pruning alone degrades accuracy, whereas combining them removes most of the penalty, and the temporal lever improves existing DVLM token pruning methods as well. On LLaDA-V-8B at a 10% retention ratio, PACE retains 97.05% of the vanilla accuracy over nine benchmarks and 93.5% on the generation tasks, whose FLOPs fall to 13.6% of vanilla.

1 INTRODUCTION

Diffusion large language models (DLLMs) (Li et al., 2025; Yu et al., 2025) have recently drawn attention as an alternative to autoregressive (AR) generation, owing to their capacity for parallel decoding (Gong et al., 2025; Wu et al., 2026). A DLLM initializes the output as a fully masked sequence and iteratively unmasks it through a learned reverse process (Austin et al., 2021; Lou et al., 2023; Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025), in which every position attends bidirectionally to the entire sequence and several masked positions are committed at each denoising step. Pioneering DLLMs such as LLaDA (Nie et al., 2026) and Dream (Ye et al., 2025), and their multimodal extensions like LLaDA-V (You et al., 2026) and LaViDa (Li et al., 2026), which we refer to as diffusion vision-language models (DVLMs), have achieved performance comparable to their AR counterparts.

In practice, however, bidirectional attention voids the KV caching that makes AR decoding efficient. Recent works (Wu et al., 2026; Ma et al., 2026; Liu et al., 2025) approximate caching by reusing keys and values across steps, since under bidirectional attention these are not invariant from step to step. Every denoising step attends over the entire assembled sequence of length n at a computational cost of $\mathcal{O}(n^2)$ (Vaswani et al., 2017), rather than with the number of newly generated tokens as in AR decoding. The step count is also larger than parallel decoding suggests. Unmasking several positions at once treats them as conditionally independent given the current context, an assumption that fails for adjacent tokens, so practical configurations keep the step count K comparable to the allotted answer length (Kang et al., 2026; Feng et al., 2026; Nie et al., 2026). The two effects compound, since decoding pays $\mathcal{O}(n^2)$ at each of the K steps, for $\mathcal{O}(Kn^2)$ overall, while cached AR decoding pays $\mathcal{O}(n^2)$.

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.

For DVLMs, the total inference cost is approximately $K \cdot C_{\text{fwd}}(n)$, where n is the assembled sequence length and $C_{\text{fwd}}(n)$ the cost of a single forward pass over it, and visual tokens dominate n on LLaDA-V (You et al., 2026), even though the visual input is fixed during generation. Each factor of $K \cdot C_{\text{fwd}}(n)$ admits a training-free lever, a spatial lever that reduces n by pruning visual tokens and a temporal lever that reduces K by committing more positions per step. Without token compression, reducing the step count causes accuracy degradation, so the temporal lever appears to trade accuracy for computation. Existing DVLM compression methods (Chang et al., 2026; Xu et al., 2026) pull the spatial lever alone and keep the default step count. We show that this trade is not fixed and the two levers are not separable. The step 0 pass runs on the full visual context and commits G/K of the G generated positions before pruning, so the step count sets the portion of the answer left to the pruned context. At the default step count the core content of the answer is decided on the pruned context, whereas with fewer steps most of the answer is determined before pruning.

We propose PACE, a training-free DVLM inference acceleration framework that co-designs both levers. Along the spatial axis, the selection criteria designed for AR-VLMs rest on premises that fail in DVLMs. Importance-based methods (Chen et al., 2024; Xing et al., 2024) prune by depth on the premise that visual attention collapses in the later layers, which does not hold in LLaDA-V (You et al., 2026). Diversity-based methods (Alvar et al., 2025; Wen et al., 2025; Zhang et al., 2026) remove redundancy by cosine similarity, which in a DVLM is dominated by a direction shared by all tokens, so near-duplicates are rare. PACE instead builds its selection signal from the queries, keys, and candidate distribution that the step 0 forward pass already materializes, at an overhead below 1% of inference cost, blending the attention of the prompt tokens with that of the still-masked response positions weighted by their probability of carrying content. Along the temporal axis, PACE selects the step count jointly with the retention ratio under a task-level rule. We evaluate two configurations, PACE-S, which pulls the spatial lever alone, and PACE-Full, which co-designs both.

Our main contributions can be summarized as follows:

- We show that the accuracy penalty of visual token compression in DVLMs is set by the share of the answer decided on the pruned context, that the step count controls this share through the step 0 commitment budget, and that committing early removes most of the penalty at aggressive budgets, which we formalize as a monotone comparative statics result.
- We propose PACE, a training-free DVLM inference acceleration framework that co-designs both levers, building its spatial signal from the attention and candidate distribution of the step 0 forward pass and setting the step count under a task-level rule.
- The temporal lever, which our exposure analysis identifies, improves existing DVLM token pruning methods, and PACE, which co-designs it with the spatial lever, retains the most accuracy at matched computation, with the margin widening as the budget tightens.

2 Analysis

All analyses are conducted under the default decoding configuration unless stated otherwise, with the detailed setup provided in the Appendix.

2.1 Preliminaries: DVLM Decoding and Its Cost

Given an image and a text prompt, the vision encoder and projector of a DVLM map the image to N_v visual tokens in the language embedding space, and its masked diffusion language model assembles the full-length sequence before decoding:

$$x^{(0)} = \big[\, p_{1:N_p} \;;\; v_{1:N_v} \;;\; \underbrace{[\mathrm{M}], \ldots, [\mathrm{M}]}_{G} \,\big], \tag{1}$$

where p denotes the prompt tokens, v the visual tokens, [M] the mask token, and G the predefined generation length. We refer to $n = N_p + N_v + G$ as the assembled length. The G masked positions are unmasked over K denoising steps. Each step runs a full bidirectional forward pass over the entire sequence at $\mathcal{O}(n^2)$ cost, and produces for every masked position a candidate token and its confidence. Under low-confidence remasking (Chang et al., 2022; Nie et al., 2026), the G/K highest-confidence candidates are committed, while the rest remain masked for the next step. A

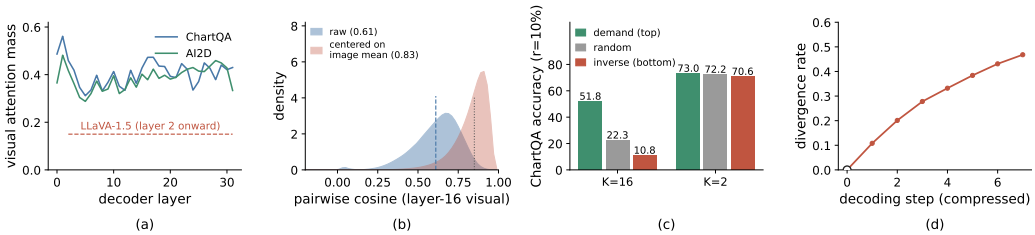


Figure 1: Failure of AR premises and divergence under compression on LLaDA-V-8B. (a) Visual attention share by decoder layer on ChartQA and AI2D, never below 0.28, against under 0.15 for LLaVA-1.5 beyond layer 2. (b) Pairwise cosine of layer-16 visual states on ChartQA, raw and centered on the image mean, with the 0.85 merge gate marked. (c) ChartQA accuracy at $r{=}10\%$ keeping the highest-demand, random, or lowest-demand tokens, at the default $K{=}16$ and $K{=}2$. (d) Divergence from the uncompressed run on ChartQA at $r{=}10\%$.

committed token is never revised, so any error at the moment of commitment is absorbed into the context of remaining steps. Total inference cost is $\mathrm{Cost} \approx K_a \cdot C_{\mathrm{fwd}}(n)$, where $C_{\mathrm{fwd}}(n)$ is the cost of a full-length forward pass. K_a is the number of steps actually taken, which falls below K when a stop token and every position before it are committed. Since visual tokens dominate n and compression can reduce C_{fwd} only from the second pass onward, the full-length first pass places a lower bound of about $1/K_a$ on the total cost. A spatial lever reduces n by pruning visual tokens, while a temporal lever reduces K by committing more positions per step. The spatial lever requires a criterion for what to prune. Existing DVLM methods (Chang et al., 2026; Xu et al., 2026) score tokens by attention at step 0 or at a shallow layer in every step, and the criteria developed for AR-VLMs are the other candidates, whose premises we examine next.

2.2 Transferability Failure of AR Premises

Importance-based selection. These approaches (Chen et al., 2024; Xing et al., 2024) rest on the premise that visual attention concentrates in the early layers and collapses thereafter, which motivates pruning by layer depth. In LLaDA-V this collapse does not occur. As shown in Fig. 1 (a), the share of attention that response positions allocate to visual tokens never falls below 0.28 in any layer on ChartQA (Masry et al., 2022) or AI2D (Kembhavi et al., 2016) and averages above 0.40 over layers 16 to 31, at step 0 and at the middle of generation, whereas FastV (Chen et al., 2024) reports under 0.15 for all visual tokens beyond layer 2 in LLaVA-1.5 (Liu et al., 2024a), where visual tokens are likewise about 90% of the keys.

Diversity-based selection. These approaches (Alvar et al., 2025; Wen et al., 2025; Zhang et al., 2026) rely on the premise that cosine similarity separates redundant tokens from informative ones, so that a covering subset preserves the visual information. We examined whether the similarity has that structure and whether coverage alone drives accuracy. First, as shown in Fig. 1 (b), over the layer-16 hidden states of the ChartQA test images the mean pairwise similarity is 0.61 and 1.8% of the pairs exceed 0.85, so near-duplicates are rare. Most of this similarity comes from a single direction shared by all tokens, to which each token has a cosine of 0.79. Centering removes it but introduces another. About 1% of the tokens have norms an order of magnitude above the median and account for 86% of the image mean, so subtracting the mean adds their direction to every other token and raises the similarity to 0.83, an artifact of these outlier tokens rather than evidence of redundancy. Second, we sorted visual tokens by the attention they receive from the response positions, which we refer to as demand, and kept the top-ranked tokens, a random subset that preserves coverage, or the bottom-ranked tokens. As shown in Fig. 1 (c), at $r{=}10\%$ and the default $K{=}16$ the three choices yield 51.8, 22.3, and 10.8, so coverage alone retains little and demand carries information in both directions, and at $K{=}2$ the spread shrinks to 73.0, 72.2, and 70.6, so the value of the selection signal grows with the share of the answer that the pruned context decides.

Neither premise holds as stated in a DVLM. Visual attention does not collapse, near-duplicates are rare, and coverage alone does not retain accuracy. A diversity term can be advantageous on some tasks, as shown in the Appendix, but it loses more on the document tasks than it gains elsewhere, so we do not adopt it. A criterion for a DVLM has to be derived from the step 0 forward pass, the only point at which all visual tokens can be scored. Every criterion imported from AR-VLMs shares an

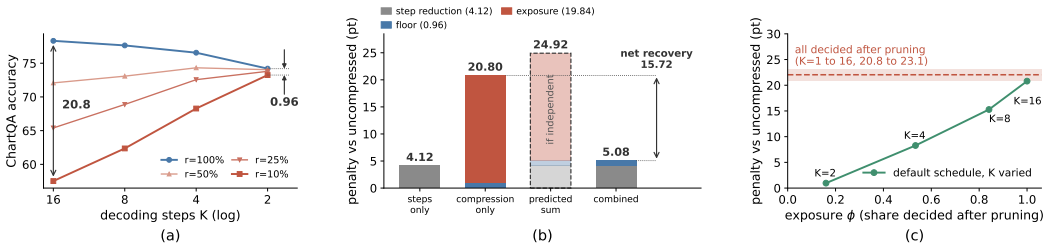


Figure 2: The compression-exposure interaction on ChartQA. (a) Accuracy as steps are removed at four retention ratios. The uncompressed run degrades while the compressed runs recover, closing the 20.8-point gap at K=16 to 0.96 at K=2 for r=10%. (b) Penalties against the uncompressed default at r=10%. Adding the two predicts 24.92 for the combination, whereas the measured loss is 5.08. (c) Penalty against exposure at r=10%, the share of answer tokens committed after pruning. The band marks runs decoded on the pruned context for K from 1 to 16.

implicit premise, that the pruned context decides the whole answer, whereas in a DVLM that share is set by the step count, a decoding hyperparameter.

2.3 The Compression-Exposure Interaction

Compression is applied after step 0, so the G/K tokens committed at step 0 are decided on the full context and every later commitment is made on the compressed one. We refer to the share of the answer decided after pruning as the *exposure*, denoted as ϕ. We show that the default schedule leaves exposure complete, that exposure accounts for most of the compression penalty, and that choosing K jointly with the retention ratio removes most of that penalty.

Step reduction under compression. To test whether the trade of accuracy for steps holds under compression, we measured accuracy over $K \in \{16, 8, 4, 2\}$ and $r \in \{100\%, 50\%, 25\%, 10\%\}$ on ChartQA (Masry et al., 2022), where G=16 at the default K=16, keeping the top fraction r of visual tokens under the step 0 selection signal. As shown in Fig. 2 (a), halving the step count from K=16 to K=8 lowers accuracy by 0.68 at r=100% but raises it by 1.00, 3.48, and 4.84 at r=50%, 25%, and 10%. At r=10%, accuracy recovers from 57.52 at K=16 to 73.24 at K=2, while uncompressed accuracy degrades from 78.32 to 74.20. Unlike the uncompressed run, where fewer steps cost accuracy, reducing steps under compression recovers accuracy, and the recovery increases with compression strength. The same reversal appears on DocVQA (Mathew et al., 2021) at r=10% and with the RedVTP (Xu et al., 2026) signal, with detailed results provided in the Appendix.

Penalty decomposition. To separate the information removed by compression from the part of the penalty that exposure controls, we compared the two levers on ChartQA at r=10%, separately and combined, against the uncompressed run. As shown in Fig. 2 (b), step reduction alone incurs 4.12 points and compression alone at the default step count incurs 20.80, whereas the combination incurs 5.08, which is the 4.12 of step reduction plus a residual of 0.96. To measure what compression removes when every answer token is decided after pruning, we discarded the step 0 commitments and re-decoded on the pruned context. The penalty is then 22.76 at K=1, 23.08 at K=2, 22.48 at K=4, 21.08 at K=8, and 20.80 at K=16. Since this penalty is nearly flat in the number of compressed rounds and coincides with the default-schedule penalty at K=16, pruning 90% of the visual tokens removes about 22 points whenever the answer is decided on the pruned context. At K=2, where the step 0 pass commits most of the answer before pruning, the added penalty falls below one point. GQA (Hudson & Manning, 2019) shows the same trend, as reported in the Appendix.

What step 0 commits. To determine which part of the answer the step 0 pass commits before pruning at each K, we traced every commitment on the ChartQA test set. At K=16 the single step 0 commitment falls on the last position of the generation block in every sample, a slot beyond the answer, so no answer token is decided before pruning. At K=2 the eight step 0 commitments cover the leading answer tokens and the terminator, deciding 84.0% of the answer. They coincide with the uncompressed output in 88.8% of cases, against 39.0% for the tokens left to the compressed pass, which therefore decides the tokens the model was uncertain about, with the committed part of the answer as textual context. Fig. 2 (c) plots exposure against the penalty, which rises with exposure from K=2 to the value at complete exposure at K=16. The step count acts on the penalty through

exposure, since fixing eight commitments at step 0 and spreading the rest over the remaining steps gives the same accuracy at $K{=}16$, 4, and 2, within 0.3 points. Errors on the pruned context propagate within an answer, as shown in Fig. 1 (d), but do not add to the penalty beyond the first error, since the penalty at complete exposure is nearly flat in the number of rounds.

Formalization. Let ε be the accuracy loss at complete exposure, non-decreasing in compression strength with $\varepsilon{=}0$ when uncompressed, and $\phi(K)$ the exposure, non-decreasing in K because the step 0 pass commits G/K tokens. We decompose accuracy as $\mathrm{Acc}(K, \varepsilon) = \mathrm{Base}(K) - \varepsilon\, g(K)$, where $\mathrm{Base}(K)$ is the uncompressed accuracy at K steps and g is a non-decreasing function of the exposure $\phi(K)$ with $g{=}1$ at complete exposure. The penalty at complete exposure is nearly flat in K, so compression enters only through exposure, and the penalty divided by its value at $K{=}16$ is nearly the same at every budget, 0.03 to 0.05 at $K{=}2$, 0.31 to 0.40 at $K{=}4$, and 0.68 to 0.74 at $K{=}8$, which is the product form with g read off directly. Since $\varepsilon\, g(K)$ is a product of non-decreasing functions, Acc has decreasing differences in (K, ε) (Topkis, 1978; Milgrom & Shannon, 1994), so the marginal value of an additional step is non-increasing in compression strength.

Proposition 1. *The set of maximizers* $\arg\max_K \mathrm{Acc}(K, \varepsilon)$ *is non-increasing in* ε *in the strong set order.*

That is, the optimal step count does not increase with compression strength. A corollary gives the compression strength beyond which removing a step pays off, $\bar{\varepsilon} = \Delta\mathrm{Base}/\Delta g$, the loss at complete exposure at which the accuracy regained through exposure equals the accuracy lost through parallel decoding. On ChartQA the thresholds predict the sign of every step reduction at all three budgets, including the loss from the last reduction to $K{=}2$ at $r{=}50\%$. Proofs and the measured thresholds are provided in the Appendix.

Complementarity of the two levers. The contribution of each lever depends on the answer length. On multiple-choice tasks the answer is a single token, but the default two-step schedule commits the terminator at step 0 and leaves the answer to the pruned pass, so exposure is complete. A single full-context pass commits both positions before pruning and matches the default schedule at half the cost, so the temporal lever alone is sufficient. On short-answer generation the step 0 pass at $K{=}2$ commits most of the answer before compression and the remaining penalty is small, so both levers apply. On longer-answer generation the optimal compressed schedule keeps at least eight steps, so the temporal lever is limited to halving the default and the spatial lever provides most of the saving. We turn these observations into three design commitments.

3 PACE: Prune Aggressively, Commit Early

We propose PACE, a training-free inference acceleration framework built on three design commitments, a step 0 selection signal, a step count chosen jointly with the retention ratio, and a task-level rule that decides which lever to pull. As shown in Fig. 3, **PACE-S** pulls the spatial lever, and **PACE-Full** additionally pulls the temporal lever. PACE shares the step 0 attention signal with RedVTP (Xu et al., 2026) and refines it with content weighting, a prompt anchor, and a gated merge. PACE recomputes the $(G + N_p) \times n$ attention rows it needs from the step 0 queries and keys, under 1% of inference cost, so it operates under the FlashAttention backend (Dao et al., 2022; Dao, 2024).

3.1 Spatial Lever: Demand-Refined Selection

Demand signal. Although the G response positions are masked at the step 0 forward pass, their attention to the image indicates which visual tokens the answer will attend to. Let A_{ij} be the attention mass from response position i to visual token j, averaged over attention heads and accumulated over all decoder layers of the step 0 pass. Summing A_{ij} over i yields the raw demand of token j.

Content weighting. When the answer is shorter than the generation block, the remaining response positions decode to filler tokens, such as the end-of-sequence, end-of-turn, pad, and stop-sequence tokens, rather than to content. Since these rows dilute the signal, PACE weights each row by its probability of carrying content from the step 0 candidate distribution:

$$w_j^{\mathrm{mask}} = \sum_{i \in \mathcal{R}} \rho_i A_{ij}, \;\; \rho_i = \max\big(0,\, 1 - P_i(\mathrm{filler})\big), \tag{2}$$

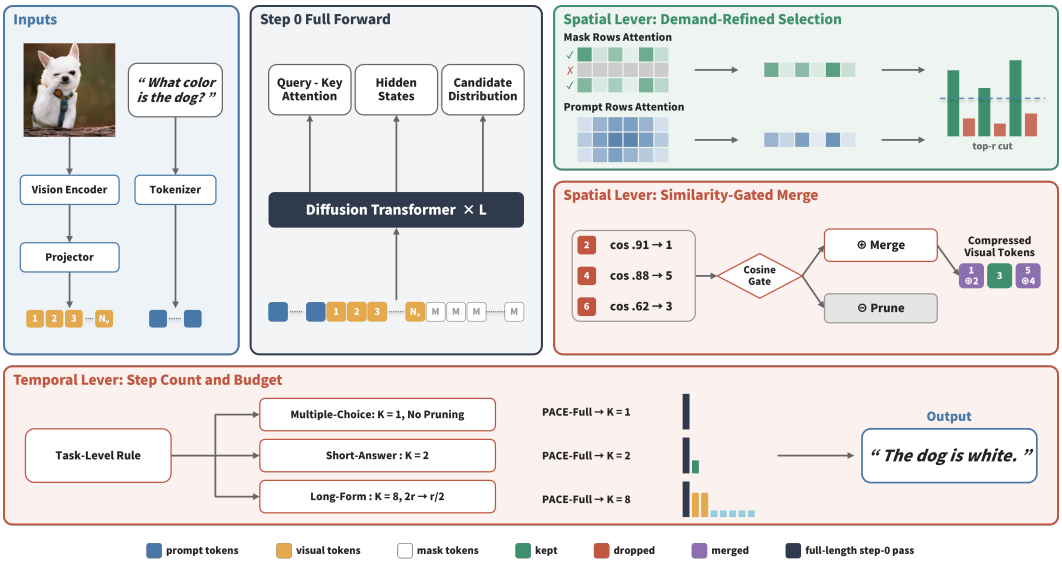


Figure 3: Overview of PACE. The spatial lever ranks visual tokens by content-weighted mask attention blended with prompt attention, keeps the top fraction r, and merges near-duplicates under a similarity gate. The temporal lever sets the step count jointly with the retention ratio by a task-level rule. Every signal is computed from the step 0 forward pass at negligible overhead. PACE-S employs the spatial lever alone at the default step count, and PACE-Full adds the temporal lever.

where $\mathcal{R}$ is the set of response positions and $P_i(\text{filler})$ the probability that the step 0 candidate distribution assigns to the filler tokens at position i. The attention and the candidate distribution are obtained from the step 0 pass before its commitment, so the weighting does not depend on which positions that pass will commit, and every position contributes according to its content probability.

Prompt anchor. The attention from the text prompt tokens to the image depends only on the question, not on what the model plans to generate. PACE normalizes both signals to unit sum over visual tokens, yielding $\hat{w}_j^{\text{mask}}$ and $\hat{w}_j^{\text{prompt}}$, then blends them into $\tilde{w}_j$:

$$\tilde{w}_j = (1-\beta)\,\hat{w}_j^{\text{mask}} + \beta\,\hat{w}_j^{\text{prompt}}, \tag{3}$$

where w_j^{prompt} is defined analogously to w_j^{mask} with the sum taken over prompt positions. PACE restricts selection to the top-$2rN_v$ tokens under the prompt signal, so that the mask signal reorders candidates that the prompt finds relevant and stays within that set. The prompt signal is fixed before any commitment and free of exposure, so PACE anchors the blend on it and lets the demand signal refine the ranking.

Selection and merge. PACE keeps the top-rN_v tokens by $\tilde{w}_j$. A dropped token is merged into its most similar kept token only if their cosine similarity over the layer-16 hidden states exceeds 0.85, a gate that admits 1.8% of the pairs, and is discarded otherwise. The kept embedding is updated as $(1-\lambda)\,e_{\text{kept}} + \lambda\,\bar{e}_{\text{drop}}$ with λ=0.25, where e denotes the input embedding and $\bar{e}_{\text{drop}}$ the mean embedding of the dropped tokens assigned to that kept token. Merging near-duplicates only keeps unrelated content out of the kept tokens, and λ constrains the change to a kept token.

3.2 Temporal Lever: Committing Early

We follow Proposition 1 to design the temporal lever of PACE-Full. Since the optimal step count is non-increasing in compression strength, a compressed run should commit earlier than the default schedule, and PACE-Full sets K jointly with r. On multiple-choice tasks the default two-step schedule decides the answer token on the pruned context, so PACE-Full runs a full-context pass, K=1, which commits both positions before pruning, and prunes nothing, since no pass remains to shorten. On short-answer generation it sets K=2, at which the step 0 pass commits most of the answer before pruning and leaves only a small penalty.

On longer-answer generation the optimal compressed schedule keeps $K \geq 8$, so step reduction is limited to halving the default, and PACE-Full additionally redistributes the token budget over time. The first $k_0 - 1$ compressed steps run with $2r$ and the remaining steps with $r/2$, where k_0 is the step at which the budget switches. The early commitments are thus made on a richer context, whereas the later steps are conditioned on the text already committed and depend less on the image, so a lower budget suffices for them. The switch point k_0 follows from cost neutrality with a constant-r run. On the generation families PACE-Full targets $r \leq 25\%$. At $r{=}50\%$ the loss at complete exposure is under 1.5 points on three of the four generation tasks, so step reduction has little to recover and lowers accuracy there, and the family rule keeps the default step count, reducing PACE-Full to PACE-S.

Task-level rule. The contribution of each lever depends on the answer length, so PACE fixes its configuration per answer-length family rather than per task. For each family we evaluated the candidate step counts $\{K_{\text{default}}, K_{\text{default}}/2, 2\}$ on a 500-sample subset of ChartQA for short answers and DocVQA for longer answers, and applied the choice to every task of the family. On LLaDA-V the rule yields $K{=}1$ for multiple-choice, where the candidate set contains the single pass, $K{=}2$ for short-answer, and $K{=}8$ with the front-loaded schedule for longer-answer.

4 Experiments

4.1 Experimental Setup

We evaluated LLaDA-V-8B (You et al., 2026) and LaViDa-8B (Li et al., 2026) on nine multimodal benchmarks, five multiple-choice (AI2D (Kembhavi et al., 2016), MMBench (Liu et al., 2024b), MME (Fu et al., 2026), ScienceQA (Lu et al., 2022), RealWorldQA (xAI, 2024)) and four generation (ChartQA (Masry et al., 2022), GQA (Hudson & Manning, 2019), DocVQA (Mathew et al., 2021), InfoVQA (Mathew et al., 2022)), under the default decoding configuration of each model and the lmms-eval protocol (Zhang et al., 2025), with MME reporting the perception score. D^{3}ToM (Chang et al., 2026) and RedVTP (Xu et al., 2026) are based on their released code. All reported numbers, including those of the vanilla models, were measured under the same protocol. All constants and the task-level rule of PACE are left unchanged on both models.

4.2 Main Results

Table 1 compares PACE with prior token compression methods. Following RedVTP, we report $r{=}75\%$ for LaViDa, whose visual tokens are already reduced by $2{\times}2$ pooling. On LLaDA-V-8B, while retaining 50% of the visual tokens, PACE-S preserves 98.41% of the vanilla performance, and its margin over baselines widens as the budget tightens. Both D^{3}ToM variants fail on the generation tasks at the default schedule. The exposure analysis accounts for where every method loses accuracy. At the default schedule the loss concentrates on the generation tasks, where the answer is committed on the pruned context. Pulling the temporal lever yields the largest improvement. On the multiple-choice tasks the rule selects a single full-context pass without pruning, so the gain belongs to the step count alone. PACE-Full reaches 93.5% and 95.5% of the vanilla accuracy at $r{=}10\%$ and 25% on the generation tasks, where both levers apply, against 84.0% and 92.1% for RedVTP, and 97.05% and 97.93% over all nine tasks. The temporal lever recovers most of the loss of RedVTP and D^{3}ToM, as Sec. 2.3 predicts. On LaViDa-8B at $r{=}75\%$, PACE-S retains 93.55% of the vanilla performance, higher than every baseline, and leads on five of the nine benchmarks. PACE applies the spatial lever alone on LaViDa, whose released schedule commits filler positions at step 0 and thus leaves exposure complete at every step count, the regime in which the analysis anticipates no recovery from step reduction. These results demonstrate the effectiveness of the co-design of spatial and temporal levers of PACE, and the transfer of the spatial lever to a second backbone.

4.3 Efficiency Analysis

As shown in Table 2, on multiple-choice tasks every method pays the full-length step 0 pass. PACE-Full removes the second pass instead of shortening it, so a single full-context pass costs 47.5 TFLOPs and runs $1.82\times$ faster than vanilla, against $1.71\times$ for PACE-S at $r{=}10\%$, while staying within 0.5% of the vanilla accuracy. On generation tasks the passes after the first dominate the cost, and the

Table 1: Accuracy on nine benchmarks for LLaDA-V-8B and LaViDa-8B. MME is the perception score, and Avg Rel is the mean accuracy relative to the vanilla model over the nine tasks. On the multiple-choice tasks PACE-Full runs a single full-context pass without pruning (†). Rows marked "+ temporal lever" apply the step counts of PACE-Full on the generation tasks.

Method	r	AI2D	MMB	MME	SQA	RWQA	ChartQA	GQA	DocVQA	InfoVQA	Avg Rel ↑
LLaDA-V-8B											
Vanilla	100%	77.72	83.08	1508.0	89.79	63.27	78.32	53.44	84.03	66.31	100.0
D^{3}ToM	50%	69.56	73.28	1382.7	80.17	52.81	32.88	49.15	42.17	41.23	76.50
D^{3}ToM-t	50%	65.97	69.59	1270.7	76.95	47.84	49.28	50.05	51.09	47.06	78.06
RedVTP	50%	76.46	82.22	1500.1	**89.19**	63.14	**72.60**	52.91	**83.49**	**65.13**	98.36
PACE-S (Ours)	50%	**76.59**	**82.39**	**1517.8**	88.60	**63.40**	72.08	**53.15**	83.21	64.94	98.41
PACE-Full (Ours)	50%	77.91†	82.99†	1500.8†	89.74†	63.14†	72.08	**53.15**	83.21	64.94	**98.65**
D^{3}ToM	25%	65.80	66.07	1139.2	75.21	47.19	12.76	44.08	24.92	30.26	63.57
+ temporal lever	25%	65.80	66.07	1139.2	75.21	47.19	71.20	49.79	41.03	36.35	76.20
D^{3}ToM-t	25%	65.84	63.23	1030.7	76.85	46.41	12.92	45.18	25.85	30.41	62.86
RedVTP	25%	73.45	77.84	1406.1	**86.17**	59.87	63.56	**52.00**	81.59	61.64	93.39
+ temporal lever	25%	73.45	77.84	1406.1	**86.17**	59.87	**73.80**	51.57	81.32	60.16	94.47
PACE-S (Ours)	25%	**73.48**	**78.01**	**1412.3**	85.52	**61.83**	65.40	51.86	81.50	60.82	93.81
PACE-Full (Ours)	25%	77.91†	82.99†	1500.8†	89.74†	63.14†	**73.80**	51.66	**81.76**	**62.18**	**97.93**
D^{3}ToM	10%	65.28	56.27	894.4	76.60	46.54	10.76	37.61	20.81	27.96	57.88
+ temporal lever	10%	65.28	56.27	894.4	76.60	46.54	70.76	48.90	38.24	35.82	72.36
D^{3}ToM-t	10%	65.54	50.52	836.1	75.95	45.49	10.56	38.47	20.76	28.00	56.61
RedVTP	10%	68.81	66.58	1171.9	**81.90**	55.82	51.80	48.04	77.71	58.02	84.65
+ temporal lever	10%	68.81	66.58	1171.9	**81.90**	55.82	72.96	**51.26**	79.26	57.71	88.47
PACE-S (Ours)	10%	**69.98**	**68.73**	**1193.5**	**81.90**	**56.86**	57.52	48.62	78.59	58.31	86.54
PACE-Full (Ours)	10%	77.91†	82.99†	1500.8†	89.74†	63.14†	**73.24**	51.08	**80.35**	**59.25**	**97.05**
LaViDa-8B											
Vanilla	100%	69.24	72.25	1353.9	72.34	54.12	67.04	56.58	63.65	36.17	100.0
D^{3}ToM	75%	67.29	69.76	1288.6	**72.73**	51.24	38.36	55.70	40.61	27.74	86.70
D^{3}ToM-t	75%	68.10	**71.74**	1319.2	71.99	52.42	43.88	**56.16**	50.24	**32.19**	91.57
RedVTP	75%	67.81	70.10	1327.5	71.84	52.55	52.32	55.84	53.31	30.60	92.72
PACE-S (Ours)	75%	**68.13**	71.31	**1327.9**	71.89	**53.07**	**53.12**	55.85	**53.86**	31.39	**93.55**

temporal lever compounds the saving. At r=10%, PACE-Full runs the four generation tasks at 13.6% of the vanilla FLOPs with a 6.47× speedup and 21.0 tokens per second, against 4.01× for RedVTP at the same budget under FlashAttention kernel. Since PACE-S and RedVTP share the cost structure, their efficiency nearly coincides, and the remaining latency gap comes from the scoring pass of RedVTP, which reads the attention of all 32 layers. D^{3}ToM re-encodes its shallow layers at full length in every step and saves the least at every budget. These results show the efficiency of PACE.

4.4 Ablation Studies

Effect of the two levers. We tested whether the gain of PACE-Full comes from combining the two levers or from either one alone. Table 3 shows that most of the recovered accuracy comes from the temporal lever, as the exposure analysis predicts, and that the spatial lever is what makes the reduced schedule cheap. Step reduction alone leaves 64.0 and 67.2 TFLOPs at K=2, since the passes still run over the full visual context, and pruning alone at the default step count stops at 89 to 91 TFLOPs on ChartQA and loses 21 to 27 points there. A single full pass at K=1 is the cheapest configuration, and PACE-Full matches it within 3 TFLOPs while recovering the accuracy that the single pass loses on GQA. Applying K=2 to RedVTP and D^{3}ToM recovers 72.96 and 70.76 on ChartQA and 51.26 and 48.90 on GQA, within 3 points of PACE-Full and 0.18 above it for RedVTP on GQA. These results demonstrate that the accuracy of PACE-Full comes from committing early on the full context, that the spatial lever adds the remaining cost saving at a penalty below one point, and that the selection signal matters only for the part of the answer decided on the pruned context.

Spatial lever analysis. We examined the three components of the spatial lever, the prompt anchor weight, the merge, and the candidate gate on ChartQA and AI2D, as shown in Table 4. Accuracy is flat between β=0.5 and 1.0, within one point on both tasks, and drops at β=0.15, where the mask rows, mostly filler and noisy at step 0, dominate the blend, by 1.5 and 1.9 points on ChartQA and by less than 0.7 on AI2D, so anchoring the blend on the prompt side matters while the exact weight does not. Every change to the merge or the candidate gate keeps accuracy within 0.6 points of the default. Turning the similarity gate off costs about 0.4 points on ChartQA at both budgets, since it merges dissimilar tokens into the kept ones, and turning the candidate gate off changes accuracy by

Table 2: Efficiency analysis on LLaDA-V-8B, averaged over the five multiple-choice and the four generation tasks. Latency is the end-to-end time per sample at batch size 1 over the first 500 samples of each benchmark and speedup the geometric mean of the latency ratio against the vanilla model. TFLOPs and latency include the vision encoder. On the multiple-choice tasks PACE-Full runs a single full-context pass without pruning (†).

Method	r	Multiple choice (5 tasks) TFLOPs ↓	Latency (s) ↓	Speedup (×) ↑	Tok/s ↑	Generation (4 tasks) TFLOPs ↓	Latency (s) ↓	Speedup (×) ↑	Tok/s ↑
Vanilla	100%	90.7	0.808	1.00	2.7	640.7	5.616	1.00	3.1
D^3ToM	50%	71.6	0.612	1.31	3.6	386.2	3.193	1.70	5.4
RedVTP		**68.9**	0.646	1.24	3.4	**343.7**	2.910	1.86	5.9
PACE-S (Ours)		69.0	**0.592**	**1.36**	**3.7**	343.8	**2.838**	**1.92**	**6.1**
PACE-Full (Ours)		47.5†	0.447†	1.82†	5.0†	343.8	**2.838**	**1.92**	**6.1**
D^3ToM	25%	62.8	0.546	1.47	4.0	267.5	2.296	2.33	7.4
RedVTP		**58.8**	0.562	1.43	3.9	208.4	1.883	2.83	9.0
PACE-S (Ours)		58.9	**0.513**	**1.57**	**4.2**	208.8	1.796	2.96	9.5
PACE-Full (Ours)		47.5†	0.447†	1.82†	5.0†	**115.2**	**1.005**	**5.56**	**17.4**
D^3ToM	10%	57.7	0.508	1.58	4.3	197.4	1.743	3.01	9.7
RedVTP		**52.9**	0.514	1.56	4.2	131.9	1.273	4.01	13.3
PACE-S (Ours)		53.0	**0.471**	**1.71**	**4.6**	131.4	1.179	4.34	14.3
PACE-Full (Ours)		47.5†	0.447†	1.82†	5.0†	**87.2**	**0.806**	**6.47**	**21.0**

Table 3: Two levers of PACE on LLaDA-V-8B at r=10%. S and T mark the spatial and temporal lever. Vanilla uses r=100% and the last row r=5%. Accuracy is on the full test set and TFLOPs on the first 500 samples.

Configuration	Lever S	Lever T	ChartQA Acc ↑	ChartQA TFLOPs ↓	GQA Acc ↑	GQA TFLOPs ↓
Vanilla, K=16			78.32	335.6	53.44	261.6
Vanilla, K=8		✓	77.64	178.1	53.40	160.7
Vanilla, K=4		✓	76.56	106.5	52.62	106.7
Vanilla, K=2		✓	74.20	64.0	51.64	67.2
Vanilla, K=1		✓	72.84	52.9	48.39	46.3
RedVTP, K=16	✓		51.80	91.3	48.04	72.9
PACE-S, K=16	✓		57.52	89.3	48.62	72.7
RedVTP, K=8	✓	✓	58.60	69.5	47.94	59.7
PACE-S, K=8	✓	✓	62.36	69.2	48.58	59.8
D^3ToM, K=2	✓	✓	70.76	55.4	48.90	51.1
RedVTP, K=2	✓	✓	72.96	54.2	**51.26**	48.8
PACE-Full (Ours), K=2	✓	✓	**73.24**	54.3	51.08	48.9
PACE-Full, r=5%	✓	✓	72.20	53.8	49.94	48.0

Table 4: Ablations of PACE-S on LLaDA-V-8B at r=25% and 10%. The default uses β=0.7, a gate of 0.85, and λ=0.25.

Setting	ChartQA 25%	ChartQA 10%	AI2D 25%	AI2D 10%
PACE-S (default)	**65.40**	**57.52**	**73.48**	**69.98**
Prompt anchor weight				
β=0.15	63.92	55.60	73.12	69.33
β=0.5	64.84	57.12	73.02	69.17
β=1.0	65.44	56.72	73.96	69.33
Merge				
gate off	65.04	57.12	74.00	69.95
λ=0.1	64.88	57.36	73.28	69.69
λ=0.5	65.36	57.32	73.06	70.08
Candidate gate				
off	65.48	57.40	73.45	70.37

at most 0.4 points. These results verify that the performance of PACE-S comes from the selection signal, that the prompt anchor is the constant that has to be set on the prompt side, and that the merge and the candidate gate change nothing at these budgets and are kept as conservative steps.

5 Conclusion

In this paper, we study inference acceleration for diffusion vision-language models, whose decoding re-encodes a fixed visual context at every denoising step, so that the cost is the product of the step count and the cost of a full-sequence pass. We showed that the two levers on this cost, pruning visual tokens and reducing the step count, are not separable. Compression is applied after the first step, and the step count sets the part of the answer that step commits on the full context, so the accuracy penalty is set by the share of the answer decided on the pruned context. Under the default schedule that share is complete and aggressive compression loses most of its accuracy, whereas committing early leaves only a small residual, a regularity we measured on the full test sets and formalized as a monotone comparative statics result. Built on this interaction, we propose PACE, a training-free DVLM inference acceleration framework, which scores visual tokens from the byproducts of the first forward pass and sets the step count jointly with the retention ratio under a task-level rule. PACE retains more accuracy than prior compression methods at lower cost on LLaDA-V-8B and LaViDa-8B and the temporal lever we identify improves existing methods as well. We believe that this work provides a new perspective on accelerating DVLM inference.

AI USE STATEMENT

In this work, we leveraged generative AI tools for writing assistance, including drafting, editing, and shortening text from the authors' outlines and results, and for code assistance, including evaluation-harness patches, analysis scripts, and plotting code. Generative AI tools also assisted in drafting analysis code, in discussing the interpretation of results and the design of follow-up experiments, and in formalizing and proving the comparative statics result of Appendix B, which the authors verified. The research problem, the method, and the final experimental design were specified and reviewed by the authors, and no experimental result was produced, altered, or selected by an AI tool. We have reviewed all AI-assisted work, and AI-edited text was checked by the authors for factual consistency with the experiments, AI-assisted code was reviewed and tested by the authors, including equivalence checks against the released implementations where applicable, and AI-assisted analyses were verified by the authors against the underlying data. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies inference efficiency of publicly released diffusion vision-language models on publicly available benchmarks. It involves no human subjects, collects no new data, and releases no new dataset. The method reduces the computation required for inference, which lowers the cost and energy footprint of deploying such models, and does not change or extend the capabilities of the underlying models. Visual token compression discards part of the visual input, so at aggressive budgets a model may miss fine-grained visual content, for example in document understanding. Deployments where such errors carry consequences should validate the chosen budget on the target task rather than rely on the benchmark averages reported here. We are not aware of any conflicts of interest related to this work.

REPRODUCIBILITY STATEMENT

All models and benchmarks used in this work are publicly available, and all evaluations were run with the lmms-eval harness under the default generation settings of each model. Baselines are run from its released code. We will release the PACE code together with the scripts for every experiment in this paper, including the ablations and the efficiency measurements.

REFERENCES

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9392–9401. IEEE, 2025.

Kazi Hasan Ibn Arif, JinYi Yoon, Dimitrios S Nikolopoulos, Hans Vandierendonck, Deepu John, and Bo Ji. Hired: Attention-guided token dropping for efficient inference of high-resolution vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1773–1781, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11315. IEEE, 2022.

Shuochen Chang, Xiaofeng Zhang, Qingyang Liu, and Li Niu. D3tom: Decider-guided dynamic token merging for accelerating diffusion mllms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 19961–19969, 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*, pp. 19–35. Springer, 2024.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Guhao Feng, Yihan Geng, Jian Guan, Wei Wu, Liwei Wang, and Di He. Theoretical benefit and limitation of diffusion language model. *Advances in Neural Information Processing Systems*, 38: 24415–24459, 2026.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *Advances in Neural Information Processing Systems*, 38, 2026.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, volume 2025, pp. 5046–5073, 2025.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 2019.

Lei Jiang, Weizhe Huang, Tongxuan Liu, Yuting Zeng, Jing Li, Lechao Cheng, and Xiaohua Xu. Fopru: Focal pruning for efficient large vision-language models. *arXiv preprint arXiv:2411.14164*, 2024.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Koo, Nam Ik Cho, et al. Parallelbench: Understanding the trade-offs of parallel decoding in diffusion llms. In *International Conference on Learning Representations*, volume 2026, pp. 89900–89938, 2026.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

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. *Advances in Neural Information Processing Systems*, 38: 105101–105134, 2026.

Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*, 2025.

Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 5334–5342, 2025.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26286–26296. IEEE, 2024a.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024b.

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. *arXiv preprint arXiv:2506.06295*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Pan Lu, Swaroop Mishra, Tanglin 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. *Advances in neural information processing systems*, 35:2507–2521, 2022.

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

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2199–2208. IEEE, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2582–2591. IEEE, 2022.

Paul Milgrom and Chris Shannon. Monotone comparative statics. *Econometrica: Journal of the Econometric Society*, pp. 157–180, 1994.

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. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations*, volume 2025, pp. 64972–65009, 2025.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22857–22867. IEEE, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

Donald M Topkis. Minimizing a submodular function on a lattice. *Operations research*, 26(2): 305–321, 1978.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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*, volume 2026, pp. 4340–4354, 2026.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9972–9991, 2025.

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*, volume 2026, pp. 57027–57051, 2026.

xAI. Grok-1.5 vision preview. `https://x.ai/news/grok-1.5v`, April 2024. Introduces the RealWorldQA benchmark. Accessed: 2026-09-19.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2783–2792, 2026.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19792–19802. IEEE, 2025.

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

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

Runpeng Yu, Qi Li, and Xinchao Wang. Discrete diffusion in large language and multimodal models: A survey. *arXiv preprint arXiv:2506.13759*, 2025.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 881–916, 2025.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in mllms. *Advances in Neural Information Processing Systems*, 38:25438–25468, 2026.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

A Related Works

A.1 Visual Token Compression in AR-VLMs

Compression of visual tokens at inference time in AR-VLMs follows two lines. Importance-based methods (Chen et al., 2024; Xing et al., 2024; Lin et al., 2025) score tokens by the attention they receive and prune on a depth schedule, on the observation that visual attention concentrates on a small subset of tokens in later layers. Text-guided variants (Zhang et al., 2024) condition the score on the prompt, while vision-side variants (Yang et al., 2025; Shang et al., 2025; Jiang et al., 2024; Arif et al., 2025) score tokens in the vision encoder and are blind to the query. Diversity-based methods (Bolya et al., 2022; Alvar et al., 2025; Wen et al., 2025; Zhang et al., 2026) take the complementary view that tokens close in embedding space are redundant, and select or merge toward a representative subset under a similarity criterion.

A.2 Efficient Inference for DLLMs

Since bidirectional attention rules out exact key-value caching, DLLM acceleration has proceeded along two lines. Caching methods reuse intermediate features across denoising steps at the cost of an approximation. Fast-dLLM (Wu et al., 2026) maintains a block-wise KV cache and recomputes it at block boundaries, dKV-Cache (Ma et al., 2026) delays reuse until a token has been decoded, and dLLM-Cache (Liu et al., 2025) refreshes prompt features at long intervals and updates the response tokens that change most between steps. Parallel decoding methods instead reduce the number of steps by unmasking more positions at once, by a confidence threshold in Fast-dLLM and by varying the degree of parallelism along the trajectory in SlowFast Sampling (Wei et al., 2026). Without such criteria, decoding more positions per step treats interdependent positions as conditionally independent and degrades quality (Kang et al., 2026; Feng et al., 2026).

For DVLMs, a third line shortens the sequence by compressing the visual tokens. RedVTP (Xu et al., 2026) scores visual tokens by the attention they receive from the still-masked response positions, since positions yet to be decoded can still affect the answer, and prunes once after the first step since these scores remain stable across steps. D^{3}ToM (Chang et al., 2026) re-selects at every step, using the tokens decoded in the previous step as deciders that define an importance map, retaining the most salient visual tokens and merging the rest into their nearest kept neighbors within a single layer, with a merge ratio that varies across steps. Both operate at the default decoding configuration and hold the step count fixed. PACE treats the sequence length and the step count as levers to be chosen together.

B Proof of the Comparative Statics Result

Setting. Let $\varepsilon \in [0, \varepsilon_{\max}]$ be the accuracy loss, in points, that compression induces when every answer token is decided after pruning, $\phi(K) \in [0,1]$ the share of the answer decided after pruning under the schedule with K steps, and $g(K) \in [0,1]$ the compression penalty at K steps divided by the penalty at complete exposure. We make two assumptions.

A1 (monotone corruption) $\varepsilon = \varepsilon(c)$ is non-decreasing in the compression strength c, which is $1 - r$ for pruning at retention ratio r, with $\varepsilon(0) = 0$.

A2 (exposure) g is non-decreasing in K. The step 0 pass commits G/K tokens on the full context, so ϕ is non-decreasing in K, and A2 amounts to the penalty being non-decreasing in ϕ.

Accuracy is $\mathrm{Acc}(K, \varepsilon) = \mathrm{Base}(K) - \varepsilon\, g(K)$, where $\mathrm{Base}(K)$ is the uncompressed accuracy at K steps. A2 is measured in Sec. 2.3. The product form assumes that ε depends only on the compression and g only on the schedule, and two measurements support this separation. When decoding is restarted after pruning, so that exposure is complete at every K, the penalty is nearly flat in K, and the K profile of the penalty normalized by its value at complete exposure is nearly the same at every retention ratio.

Lemma 1 (increasing differences). For $K' > K$ and $\varepsilon' > \varepsilon$:

$$\big[\varepsilon' g(K') - \varepsilon' g(K)\big] - \big[\varepsilon g(K') - \varepsilon g(K)\big] = (\varepsilon' - \varepsilon)\big(g(K') - g(K)\big) \;\geq\; 0,$$

so the penalty $\varepsilon\, g(K)$ has increasing differences in (K, ε) and Acc has decreasing differences. □

Proposition 1 (monotone optimal step count). The set of maximizers $K^*(\varepsilon) = \arg\max_K \mathrm{Acc}(K, \varepsilon)$ is non-increasing in ε in the strong set order. Combined with A1, the optimal step count is non-increasing in the compression strength.

Proof. Let $\varepsilon' > \varepsilon$, $K \in K^*(\varepsilon)$, and $K' \in K^*(\varepsilon')$. If $K' \leq K$ there is nothing to show, so let $K' > K$. By Lemma 1,

$$\mathrm{Acc}(K', \varepsilon') - \mathrm{Acc}(K, \varepsilon') \;\leq\; \mathrm{Acc}(K', \varepsilon) - \mathrm{Acc}(K, \varepsilon).$$

The left side is non-negative since K' is optimal at ε', so the right side is non-negative and K' is also optimal at ε. The right side is non-positive since K is optimal at ε, so the left side is zero and K is also optimal at ε'. Hence $\max(K, K') \in K^*(\varepsilon)$ and $\min(K, K') \in K^*(\varepsilon')$, which is the strong set order. Maximizers exist since K ranges over the finite set $\{1, \ldots, K_{\max}\}$, and no concavity or uniqueness assumption is needed. This is the monotone comparative statics theorem of Topkis

(1978) and Milgrom & Shannon (1994) applied to Acc on the lattice $\{1, \dots, K_{\max}\} \times [0, \varepsilon_{\max}]$. With A1, ε is non-decreasing in the compression strength, so the optimal step count is non-increasing in it. □

Corollary 1 (sign crossing of the marginal step). Fix a schedule K and add one step. The change in accuracy splits into two terms:

$$\Delta\mathrm{Acc}(K, \varepsilon) = \mathrm{Acc}(K+1, \varepsilon) - \mathrm{Acc}(K, \varepsilon) = \Delta\mathrm{Base}(K) - \varepsilon\, \Delta g(K), \tag{4}$$

where $\Delta\mathrm{Base}(K) = \mathrm{Base}(K+1) - \mathrm{Base}(K)$ is what the added step gains without compression and $\Delta g(K) = g(K+1) - g(K)$ is the share of the compression loss that the added step newly exposes, with $\Delta g(K) \geq 0$ by A2. The first term does not depend on ε and the second grows linearly in ε, so the sign of $\Delta\mathrm{Acc}$ changes at most once as the compression loss grows. Three cases cover the sign. If $\Delta\mathrm{Base}(K) \leq 0$, both terms are non-positive and the added step never raises accuracy. If $\Delta\mathrm{Base}(K) > 0$ and $\Delta g(K) = 0$, the added step raises accuracy at every ε. If $\Delta\mathrm{Base}(K) > 0$ and $\Delta g(K) > 0$, the two terms cancel at the threshold:

$$\bar{\varepsilon}(K) = \frac{\Delta\mathrm{Base}(K)}{\Delta g(K)}. \tag{5}$$

Below the threshold the added step raises accuracy, and above it the added step lowers accuracy. The crossing lies inside $[0, \varepsilon_{\max}]$ when $\bar{\varepsilon}(K) < \varepsilon_{\max}$. The same argument applies to any pair $K < K'$ with $\Delta\mathrm{Base}$ and Δg taken between K and K'.

Relation to the measurements. On ChartQA at r=10% the penalty is 0.96, 8.28, 15.28, and 20.80 at exposures 0.160, 0.534, 0.841, and 1, and 20.8 to 23.1 at complete exposure for every K. The penalty is monotone and convex in exposure, 0.96 against $22 \times 0.160 = 3.5$ at K=2, since the tokens left to the compressed pass are decided with the correct skeleton as textual context. Divided by its value at K=16, the penalty at K=2, 4, and 8 is 0.046, 0.398, and 0.735 at r=10%, 0.031, 0.310, and 0.678 at r=25%, and 0.026, 0.359, and 0.731 at r=50%, so the K profile is nearly independent of the budget, which is the product form with $g(K)$ equal to this profile. With g from the r=10% profile, the thresholds $\bar{\varepsilon} = \Delta\mathrm{Base}/\Delta g$ for halving from 16, 8, and 4 steps are 2.6, 3.2, and 6.7 points, against losses at complete exposure of 6.2, 12.9, and 20.8 at r=50%, 25%, and 10%. Every halving down to K=4 is therefore predicted to help at all three budgets, as measured, and the last halving to K=2 is predicted to lower it at r=50%, where we measured −0.28.

Limitations of the model. A1 and the separable form of the penalty are assumptions supported by the measured curves. The result guarantees the direction of the effect, and the size of the interaction and the location of the crossing are empirical.

C Algorithm

Algorithm 1 provides the spatial lever of PACE, applied once after the step 0 forward pass. PACE-S runs it at the default step count. PACE-Full runs it with the step count and budgets of Table 5. On the long-answer tasks, PACE-Full applies it a second time at step k_0, with r_2 in place of r and the kept tokens as the candidate set. All inputs are byproducts of the step 0 pass.

Table 5: Task-level rule of PACE-Full on LLaDA-V-8B. K_{def} is the default step count. On the long-answer family the algorithm runs with r_1 after step 0 and again with r_2 at step $k_0 = (K+2)/3$ rounded down.

Family	K	r_1	r_2
Multiple-choice	1	no pruning	–
Short answer	2	r	–
Long answer	$K_{\mathrm{def}}/2$	$2r$	$r/2$

D Additional Analysis Results

Attention depth profile. For Fig. 1 (a), we measured at every layer the share of attention that the response rows place on visual tokens, averaged over heads and the full test set, at step 0 and at the

Algorithm 1 Demand-Refined Selection and Gated Merge

Require: head-averaged step 0 attention $A^{(l)}$ of each decoder layer l, from response positions $\mathcal{R}$ and prompt positions $\mathcal{P}$ to the N_v visual tokens; filler probability $P_i(\text{filler})$ for $i \in \mathcal{R}$; layer-16 hidden states H of the visual tokens and their input embeddings E; retention ratio r; constants β, λ, gate τ

Ensure: kept visual tokens $E^{\text{keep}} \in \mathbb{R}^{rN_v \times d}$

$A \leftarrow \sum_l A^{(l)}$

$\rho_i \leftarrow \max(0,\, 1 - P_i(\text{filler}))$ for $i \in \mathcal{R}$

$w_j^{\text{mask}} \leftarrow \sum_{i \in \mathcal{R}} \rho_i A_{ij}, \quad w_j^{\text{prompt}} \leftarrow \sum_{i \in \mathcal{P}} A_{ij}$

$\tilde{w} \leftarrow (1-\beta)\, \hat{w}^{\text{mask}} + \beta\, \hat{w}^{\text{prompt}}, \quad \hat{w} = w / \sum_j w_j$

$\mathcal{C} \leftarrow \text{Top}(\hat{w}^{\text{prompt}},\, 2rN_v), \quad \mathcal{K} \leftarrow \text{Top}(\tilde{w}[\mathcal{C}],\, rN_v)$

$\hat{H} \leftarrow H / \|H\|_2$ row-wise; for each $d \notin \mathcal{K}$, $k^\star(d) \leftarrow \arg\max_{k \in \mathcal{K}} \hat{H}_d^\top \hat{H}_k$

$\mathcal{D}_k \leftarrow \{\, d \notin \mathcal{K} : k^\star(d) = k,\ \hat{H}_d^\top \hat{H}_k > \tau \,\}$ for $k \in \mathcal{K}$

$E_k \leftarrow (1-\lambda)\, E_k + \lambda \cdot \text{mean}_{d \in \mathcal{D}_k} E_d$ for $k \in \mathcal{K}$ with $\mathcal{D}_k \neq \emptyset$

return $E^{\text{keep}} \leftarrow E[\mathcal{K}_{\text{sorted}}]$

Table 6: Step-reduction grid on ChartQA. Accuracy on the full test set for LLaDA-V-8B over the step count K at each retention ratio. At K=1 every configuration gives 72.84, since all tokens are committed before pruning.

Configuration	K=16	K=8	K=4	K=2
Vanilla, r=100%	78.32	77.64	76.56	74.20
PACE-S, r=50%	72.08	73.08	74.32	74.04
PACE-S, r=25%	65.40	68.88	72.56	73.80
PACE-S, r=10%	57.52	62.36	68.28	73.24
RedVTP, r=10%	51.80	58.60	66.60	72.96

middle of generation. On ChartQA the per-layer share at step 0 lies between 0.311 at layer 5 and 0.561 at layer 1, and the mean over layers 16 to 31 is 0.420 at step 0 and 0.430 at mid-generation. On AI2D the range is 0.288 to 0.482 at step 0 and 0.286 to 0.513 at mid-generation, with means of 0.409 and 0.406 over layers 16 to 31. Since no layer falls below 0.28, the deep-layer collapse that importance-based criteria rely on in AR models is absent.

Homogeneity of visual embeddings. For Fig. 1 (b), we computed the mean pairwise cosine similarity among the layer-16 visual hidden states of each of the 2,500 ChartQA test images at step 0, over all tokens. The mean is 0.61 on raw states, and 1.8% of the pairs exceed the merge gate of 0.85. Each token has a cosine of 0.79 to the mean of the tokens below the 99th norm percentile, so most of the raw similarity is a shared direction. The remaining 1% of the tokens have norms 8.9 times the median and account for 86% of the per-image mean, so subtracting that mean adds their direction to every other token and raises the mean similarity to 0.83. Centering the other tokens on their own mean gives 0.01 by construction with a standard deviation of 0.17, so structure remains in the residual, but not the near-duplicate structure that a merge criterion needs.

Selection direction. For Fig. 1 (c), we kept 10% of the visual tokens on ChartQA by three rules that share the step 0 mask-row demand signal of RedVTP. At the default K=16 the highest-demand tokens give 51.80, a random subset 22.28, and the lowest-demand tokens 10.84. At K=2 the same rules give 72.96, 72.16, and 70.56, so the spread of 41 points shrinks to 2.4 once most of the answer is decided before pruning.

Step-reduction grid. Table 6 gives the ChartQA grid behind Sec. 2.3 and tests whether the recovery from step reduction depends on the selection signal or on the budget. Under compression, accuracy rises as K falls at every budget, and the rise grows with compression strength, whereas the uncompressed model loses accuracy at every halving. The same reversal appears with the RedVTP signal, and the gap between the two signals shrinks as exposure falls, from 5.72 points at K=16 to 3.76 at K=8, 1.68 at K=4, and 0.28 at K=2. On the 500-sample subset of DocVQA, halving K from 16 to 8 changes accuracy by -1.47 without compression and by -0.76, -0.26, and $+1.26$ at r=50%, 25%, and 10%, the same sign flip at the aggressive budget. The prompt anchor shows the same dependence on exposure, since on ChartQA at r=10% the margin of β=0.7 over β=0.3 is -0.04, -0.08, $+0.68$, and $+1.80$ at K=2, 4, 8, and 16. These results verify that the recovery is a property

Table 7: Accuracy with the whole schedule decoded on the pruned context at $r{=}10\%$ on LLaDA-V-8B, against the uncompressed model at the same K.

ChartQA	$K{=}1$	$K{=}2$	$K{=}4$	$K{=}8$	$K{=}16$
Uncompressed	72.84	74.20	76.56	77.64	78.32
Pruned, all decided after pruning	50.08	51.12	54.08	56.56	57.52
Penalty	22.76	23.08	22.48	21.08	20.80
GQA	$K{=}1$	$K{=}2$			
Uncompressed	48.39	51.64			
Pruned, all decided after pruning	42.89	46.35			
Penalty	5.50	5.29			

of compression rather than of a particular selection signal, and that the value of the signal and of its prompt anchor grows with the share of the answer decided on the pruned context.

Penalty at complete exposure. Table 7 tests whether the compression penalty depends on the number of compressed rounds or only on exposure. The step 0 commitments were discarded and the whole schedule was decoded on the pruned context, so every answer token is decided after pruning. The penalty is nearly flat in K on both tasks, and on ChartQA it coincides at $K{=}16$ with the default schedule, whose single step 0 commitment falls on the last slot of the generation block. On GQA, whose answers are single words, compression removes about 5.4 points at complete exposure against about 22 on ChartQA. These results demonstrate that compression enters the penalty through exposure and not through the number of rounds, which is the separable form the formalization assumes.

Commitment budget of the step 0 pass. This experiment tests whether the step count acts on the penalty through anything other than the number of tokens the step 0 pass commits. Fixing eight commitments at step 0 and spreading the rest uniformly over the other steps gives 73.44 at $K{=}16$ and 73.16 at $K{=}4$ on ChartQA and 51.22 at $K{=}16$ on GQA at $r{=}10\%$, against 73.24 and 51.08 for $K{=}2$, at 57.6 TFLOPs per sample on ChartQA against 54.3. On MMBench, whose generation block has two positions, step 0 commits the terminator with confidence near one in every traced sample and the answer letter is committed at step 1 on the pruned context, whereas a single full-context pass, $K{=}1$, commits both positions before pruning and matches the default two-step schedule within 0.5% on all five multiple-choice tasks at 47.5 against 90.7 TFLOPs. At $r{=}5\%$, PACE-Full reaches 72.20 on ChartQA and 49.94 on GQA, so the remaining penalty at $K{=}2$ grows to 2.0 and 1.7 points when 95% of the visual tokens are pruned. These results verify that the temporal lever acts through the step 0 commitment budget, that on multiple-choice tasks this budget is realized by a single pass, and that the remaining penalty stays small down to $r{=}5\%$.

Feedback traces. This analysis tests whether errors on the pruned context feed back into later commitments, as Fig. 1 (d) suggests, and whether that feedback adds to the penalty. We recorded every commitment as a tuple of step, position, token, and confidence, and paired the compressed run at $r{=}10\%$ with the uncompressed run on each of the 2,500 ChartQA test samples at the default $K{=}16$ under deterministic decoding. A commitment is divergent when the two runs fix different tokens at the same position, and only positions committed by both runs enter the statistics. The divergence rate is 0 at step 0 and 0.108, 0.201, 0.278, 0.332, 0.384, 0.431, and 0.468 at steps 1 to 7. Pooled over steps after step 0, the probability of divergence is 0.740 after an earlier divergence in the same sample and 0.168 before the first, a ratio of 4.41, and the ratio within each step ranges from 2.4 to 5.3 over steps 2 to 8. A linear probability model with step fixed effects estimates the effect of a prior divergence at $+0.520$ with $t{=}61.8$ over 14,066 commitments. Sample fixed effects drive the estimate to -0.008, the expected downward bias for a lagged state in short panels, so we rely on two null constructions. A parametric null that draws each commitment independently with a divergence probability set by its sample and step gives $+0.198$ with a 95% interval of $[0.175, 0.221]$ over 2,000 replicates, under 40% of the observed value, and a swap permutation that preserves the divergence count of every sample and every step gives $+0.462$ with a 95% interval of $[0.443, 0.508]$ over 1,000 permutations, which the observed $+0.520$ exceeds. Divergent commitments are also made with lower confidence, 0.642 against 0.784. These results demonstrate that errors on the pruned context feed back into later commitments within an answer. Since the penalty at complete exposure is nearly flat in the number of rounds, this feedback changes which tokens are wrong but not how many, which is why the main text attributes the penalty to exposure rather than to propagation.

Table 8: Per-task cost on LLaDA-V-8B. Each cell reports TFLOPs per sample and latency in seconds, measured on the first 500 samples of each benchmark under the protocol of Appendix G, with every compression method under FlashAttention. On the multiple-choice tasks PACE-Full runs a single full-context pass without pruning (†), independent of r.

Method	r	AI2D	MMB	MME	SQA	RWQA	ChartQA	GQA	DocVQA	InfoVQA
Vanilla	100%	84 / 0.73	66 / 0.56	113 / 1.04	64 / 0.55	127 / 1.16	336 / 2.91	262 / 2.21	982 / 8.69	984 / 8.65
D^3ToM	50%	66 / 0.55	52 / 0.43	89 / 0.77	51 / 0.43	100 / 0.87	227 / 1.87	167 / 1.37	577 / 4.83	574 / 4.70
RedVTP	50%	63 / 0.59	50 / 0.46	86 / 0.81	49 / 0.46	96 / 0.91	188 / 1.61	154 / 1.32	519 / 4.44	515 / 4.28
PACE-S	50%	64 / 0.53	50 / 0.42	86 / 0.74	49 / 0.42	96 / 0.84	188 / 1.54	153 / 1.26	519 / 4.34	515 / 4.21
PACE-Full	50%	43 / 0.40†	34 / 0.30†	60 / 0.58†	33 / 0.30†	67 / 0.66†	188 / 1.54	153 / 1.26	519 / 4.34	515 / 4.21
D^3ToM	25%	58 / 0.49	46 / 0.39	78 / 0.69	45 / 0.38	88 / 0.78	170 / 1.38	123 / 1.04	391 / 3.45	386 / 3.32
RedVTP	25%	54 / 0.51	43 / 0.40	73 / 0.70	42 / 0.40	82 / 0.80	124 / 1.07	103 / 0.93	307 / 2.85	300 / 2.68
PACE-S	25%	54 / 0.46	43 / 0.36	73 / 0.64	42 / 0.36	82 / 0.73	125 / 1.04	103 / 0.87	307 / 2.71	300 / 2.57
PACE-Full	25%	43 / 0.40†	34 / 0.30†	60 / 0.58†	33 / 0.30†	67 / 0.66†	56 / 0.48	52 / 0.44	180 / 1.63	173 / 1.48
D^3ToM	10%	53 / 0.46	42 / 0.36	72 / 0.64	41 / 0.36	81 / 0.73	129 / 1.12	97 / 0.84	285 / 2.58	279 / 2.44
RedVTP	10%	48 / 0.46	39 / 0.37	66 / 0.65	38 / 0.36	74 / 0.73	91 / 0.87	73 / 0.69	186 / 1.85	178 / 1.68
PACE-S	10%	49 / 0.42	39 / 0.33	66 / 0.59	38 / 0.33	74 / 0.67	89 / 0.78	73 / 0.64	186 / 1.72	178 / 1.57
PACE-Full	10%	43 / 0.40†	34 / 0.30†	60 / 0.58†	33 / 0.30†	67 / 0.66†	54 / 0.50	49 / 0.42	127 / 1.23	118 / 1.08

E Implementation Details of PACE

Selection signal. All signals are computed from the step 0 forward pass. Let $A^{(l)}$ be the head-averaged attention of decoder layer l, recomputed from the stored queries and keys for the G response rows and the N_p text prompt rows only, and let $A = \sum_l A^{(l)}$ be its accumulation over layers. For visual token j, the mask-side demand is $w_j^{\text{mask}} = \sum_{i \in \mathcal{R}} \rho_i A_{ij}$ with $\rho_i = \max(0,\, 1 - P_i(\text{filler}))$, where $\mathcal{R}$ is the set of response positions and $P_i(\text{filler})$ is the probability mass that the step 0 candidate distribution at position i places on the filler tokens, namely the end-of-sequence, end-of-turn, pad, and stop-sequence tokens. The prompt-side demand is $w_j^{\text{prompt}} = \sum_{i \in \mathcal{P}} A_{ij}$ over the text prompt positions $\mathcal{P}$. Both are normalized to unit sum over visual tokens and blended as $\tilde{w} = (1-\beta)\,\hat{w}^{\text{mask}} + \beta\,\hat{w}^{\text{prompt}}$ with $\beta = 0.7$. The candidate gate restricts selection to the $2rN_v$ tokens with the highest $\hat{w}^{\text{prompt}}$, and the rN_v tokens with the highest $\tilde{w}$ inside this set are kept.

Merge. Each dropped token is assigned to the kept token with the highest cosine similarity between their layer-16 hidden states, only if the similarity exceeds 0.85, and is discarded otherwise. The embedding of a kept token that receives $n_k \geq 1$ dropped tokens becomes $(1-\lambda)\, e_k + \lambda\, \bar{e}_{\text{drop}}$ with $\lambda = 0.25$, where $\bar{e}_{\text{drop}}$ is the mean of the assigned embeddings.

Temporal lever. On the generation families PACE-Full targets $r \leq 25\%$. The task-level rule was set once per answer-length family on a 500-sample subset of ChartQA and DocVQA by evaluating the candidate set $\{K_{\text{default}},\, K_{\text{default}}/2,\, 2\}$ at the target budget and choosing the most accurate entry, with ties within one point resolved toward the cheaper entry. The rule chose $K{=}1$ on the five multiple-choice tasks, where the candidate set contains the single pass and accuracy is within 0.5% of the default at half the cost, $K{=}2$ on ChartQA and GQA, and $K{=}8$ on DocVQA and InfoVQA. On the two long-answer tasks the token budget is front-loaded. Steps 1 to $k_0 - 1$ run at $2r$ and steps from k_0 at $r/2$, where k_0 solves the cost-neutrality equation $2r(k_0{-}1){+}(r/2)(K{-}k_0) = r(K{-}1)$, giving $k_0 = (K+2)/3$, rounded down to $k_0{=}3$ at $K{=}8$ so that the total budget, $6.5r$ against $7r$, never exceeds the budget it replaces. The narrowing at k_0 reuses the step 0 scores $\tilde{w}$.

Generation settings. We used the default generation settings of each model in lmms-eval (Zhang et al., 2025), with temperature 0, no classifier-free guidance, and low-confidence remasking. On LLaDA-V-8B (You et al., 2026), the multiple-choice tasks generate $G{=}2$ tokens in $K{=}2$ steps, ChartQA and GQA generate $G{=}16$ in $K{=}16$ steps with a newline stop sequence, and DocVQA and InfoVQA generate $G{=}32$ in $K{=}16$ steps. LaViDa-8B (Li et al., 2026) pools visual tokens 2×2 to 835 tokens and decodes with one step per position by default, 16 steps for the 16-token block on the short-answer tasks and 32 for the 32-token block on the longer-answer tasks, with all constants of PACE transferred unchanged. The overhead of the signal recomputation and the merge is under 1% of inference cost.

Table 9: Selection rule of PACE-S on LLaDA-V-8B at r=10%. Metrics follow Table 1, and Avg Rel is the mean accuracy relative to vanilla LLaDA-V over the nine tasks.

Selection rule	AI2D	MMB	MME	SQA	RWQA	ChartQA	GQA	DocVQA	InfoVQA	Avg Rel ↑
top-$\tilde{w}$ (Ours)	**69.98**	68.73	1193.5	81.90	**56.86**	57.52	48.62	**78.59**	**58.31**	**86.54**
facility location	68.65	**75.09**	**1284.8**	**84.58**	56.21	**57.84**	**49.48**	62.95	54.70	85.64

F BASELINES

RedVTP (Xu et al., 2026) and D^{3}ToM (Chang et al., 2026) are based on their released code. RedVTP scores each visual token by the step 0 attention it receives from the response rows that remain masked after the step 0 transfer, averaged over heads and layers, keeps the top fraction r, and prunes once. D^{3}ToM merges visual tokens at layer l^*=3 in every step, guided by the attention of the positions committed in the previous step, at a fixed merge ratio, and D^{3}ToM-t replaces the fixed ratio by the released adaptive schedule, whose merge ratio at step i is $\text{low} + (1 - \text{low})(1 - i/K)$ with $\text{low} = \max(0,\, 2\,\text{MR} - 1)$ for a nominal merge ratio MR.

G EFFICIENCY MEASUREMENT PROTOCOL

Hardware and protocol. All measurements ran on one NVIDIA RTX 4090 with 24 GB at batch size 1 in bfloat16 under PyTorch 2.4.1. Latency is the end-to-end time per sample including the vision encoder, averaged over the first 500 samples of each benchmark, with all configurations of a benchmark measured in one contiguous window so that they share the machine load. Re-measuring four AI2D configurations under a different load changed their latencies by at most 1.1%. FLOPs are counted per sample with the PyTorch flop counter over the same 500 samples and include the vision encoder, which contributes 2 to 20 TFLOPs per sample depending on the task. Tokens per second divide the mean number of committed tokens on each task, 2 on the multiple-choice tasks, 7.03 on ChartQA, 5.97 on GQA, and 32 on DocVQA and InfoVQA, by the latency.

Attention kernel. PACE passes no attention mask, so PyTorch scaled dot-product attention dispatches to the FlashAttention kernel (Dao et al., 2022). RedVTP (Xu et al., 2026) and D^{3}ToM (Chang et al., 2026) were measured under the same FlashAttention path as PACE, and the vanilla reference under its released code. The latency gap that remains between PACE-S and RedVTP at equal FLOPs comes from the scoring pass of the RedVTP implementation, which reads the step 0 attention of all 32 layers in single precision.

Per-task results. Table 8 gives the per-task values behind Table 2. PACE-S and RedVTP coincide in FLOPs on every task, since both prune once after step 0, and their latency gap is the scoring pass described above. D^{3}ToM costs the most at every budget because its shallow layers run at full length in every step.

H ADDITIONAL RESULTS

Selection rule. To test whether selection benefits from a diversity term, we replaced the top-$\tilde{w}$ rule of PACE-S with a demand-weighted facility-location selection at r=10% on LLaDA-V-8B (You et al., 2026), as shown in Table 9. The diversity term gains on five of the nine tasks, MMBench and MME most of all, and loses on the other four, with DocVQA and InfoVQA falling far below the top-$\tilde{w}$ rule. The losses on the document tasks outweigh the gains, lowering Avg Rel by 0.90, so we keep the single top-$\tilde{w}$ rule across tasks. The facility-location variant is insensitive to β, varying within 0.6 points on MMBench between β=0.3 and 1.0. For reference, the β curve of the top-$\tilde{w}$ rule on ChartQA at $\beta \in \{0, 0.15, 0.3, 0.5, 0.7, 1.0\}$ is 55.04, 55.60, 55.72, 57.12, 57.52, and 56.72 at r=10% and 63.36, 63.92, 64.92, 64.84, 65.40, and 65.44 at r=25%.

Compatibility with KV caching. To test whether PACE is compatible with an approximate KV cache, we ran PACE-Full on top of the block cache of Fast-dLLM (Wu et al., 2026) on LLaDA-V-8B. The step 0 pass runs on the full context and fills the cache as usual, and PACE reads its signal from that same pass. After the step 0 commitment, the sequence and the cached prefix are pruned to the kept visual tokens, with the cached keys re-rotated to their compacted positions, and every later step is a Fast-dLLM incremental pass over the response block attending to the pruned prefix cache.

Table 10: PACE-Full with and without the Fast-dLLM KV cache on LLaDA-V-8B. Accuracy on the full test sets at the step counts of Table 5. Uncached values are those of Table 1. The merge step is inactive under the cache.

Method	ChartQA	GQA	DocVQA	InfoVQA
r=25%				
PACE-Full	73.80	51.66	81.76	62.18
PACE-Full + cache	73.80	51.49	81.52	62.64
r=10%				
PACE-Full	73.24	51.08	80.35	59.25
PACE-Full + cache	73.84	51.46	80.21	60.84

On the long-answer tasks the second pruning at step k_0 is applied to the cache in the same way. The merge step is inactive under the cache, since the cached keys of the kept tokens are not re-encoded, so the cached runs use selection alone. Table 10 compares accuracy with and without the cache at the step counts of Table 5. The cache lowers accuracy by at most 0.24 points and raises it by up to 1.6, demonstrating the compatibility of PACE with KV caching. Note that on the multiple-choice tasks PACE-Full is a single full-context pass without pruning, so no cache is involved. For reference, the cache alone at K=2 without pruning gave 73.88 on ChartQA and 51.42 on GQA, against 74.20 and 51.64 uncached.

I LIMITATIONS

Our evidence comes from two DVLMs, LLaDA-V-8B (You et al., 2026) and LaViDa-8B (Li et al., 2026), that share the LLaVA-style vision stack (Liu et al., 2024a). Unified discrete-token models represent images differently and remain untested, although the ingredients of the selection signal exist there as well. All measurements use uncached decoding, the released configuration of LLaDA-V and LaViDa. Approximate key-value caches for DLLMs encode the visual context once, so they remove the passes that the spatial lever shortens. PACE is compatible with the Fast-dLLM cache with accuracy unchanged, but the FLOPs then no longer depend on the retention ratio, so the saving of the spatial lever is specific to uncached decoding. The benchmarks generate at most 32 tokens, so chain-of-thought style generation, where the base accuracy may depend on the step count more steeply, is outside our measurements, and the model of Appendix B predicts that the benefit of the temporal lever shrinks in that regime. On schedules that commit nothing at step 0, exposure is complete at every step count, so the analysis predicts no recovery from step reduction and the temporal lever only trades accuracy for cost. At the step counts the rule selects, the selection signal contributes little, since a random subset at K=2 is within 1.1 points of PACE-Full on ChartQA, so its value is concentrated where most of the answer is decided on the pruned context. The task-level rule was set once per answer-length family on the first 500 samples of ChartQA and DocVQA, and the long-answer choice follows from cost rather than from the subset. The comparative statics result guarantees the direction of the interaction and not its size.