RECAP: RECIPROCAL SPARSE ATTENTION AND HIERARCHICAL CACHING FOR EFFICIENT VIDEO DIFFUSION

Anonymous authors
Paper under double-blind review

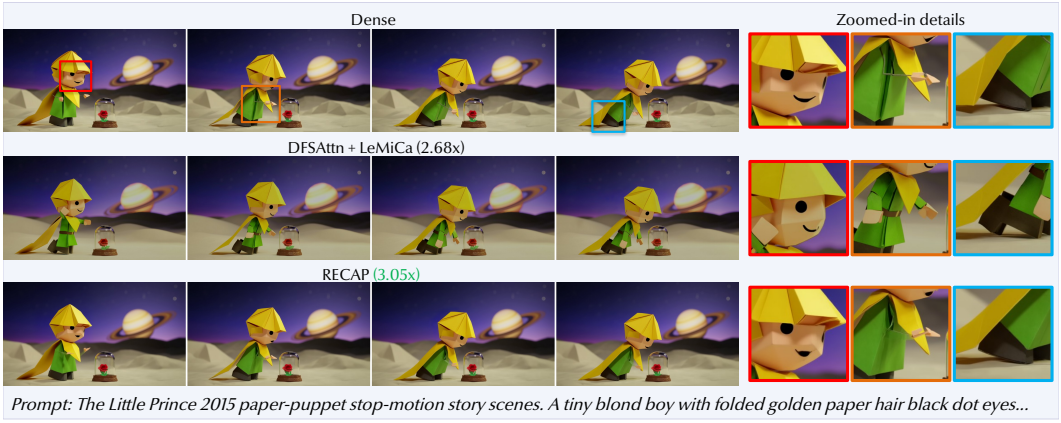


Figure 1: Sample video and results on HunyuanVideo (720p, 5s). RECAP achieves a 3.05× speedup while outperforming all tested direct sparse–cache compositions in both fidelity and speed.

ABSTRACT

Diffusion transformers generate high-quality videos but remain computationally expensive. Sparse attention and temporal caching reduce computation within and across denoising steps, respectively, offering a natural composition. However, predictions produced with sparse attention become sources for temporal reuse, coupling their errors along the sampling trajectory. We observe that improving the accuracy of these sources recovers fidelity without changing the reuse schedule, and that sparsification errors remain directionally aligned across nearby steps, suggesting that correction information can be shared rather than recomputed. These observations motivate RECAP, a training-free method that couples sparse attention with hierarchical caching through reciprocal correction, adaptively coordinated by online error feedback. Cached dense refinement corrects omitted interactions in current sparse estimates, while fresh sparse statistics track input changes and regulate the aging correction. The resulting predictions then renew the sources for whole-step prediction reuse, allowing prediction history to advance between dense refreshes. We evaluate RECAP on multiple video diffusion models against sparse-attention methods, caching methods, and their direct compositions. On HunyuanVideo at 720p 5-second generation, RECAP achieves a 3.05× speedup and outperforms all tested direct compositions in both fidelity and inference speed.

1 INTRODUCTION

Diffusion transformers have emerged as a powerful architecture for visual generation (Peebles & Xie, 2023), but the quadratic complexity of attention and repeated transformer evaluations make inference expensive. Video generation is especially costly due to the length of its spatiotemporal token sequences (Yang et al., 2025b; Kong et al., 2024). Sparse attention reduces the interactions computed within each denoising step (Xi et al., 2025; Hu et al., 2026), while temporal caching reuses features or predictions across steps (Liu et al., 2025a). Since the two approaches save computation along different axes, a natural way to accelerate further is to compose them directly. In such a

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.

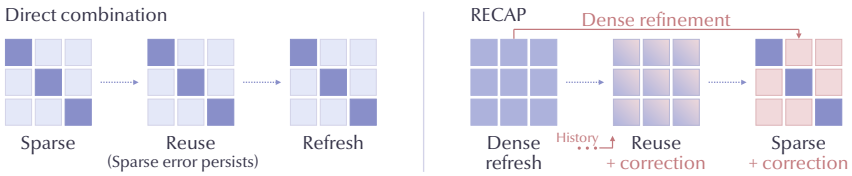


Figure 2: Motivation of RECAP. Direct composition reuses sparse outputs, so sparse error persists. RECAP caches dense refinement to correct sparse steps, whose outputs renew the reuse history.

composition, however, sparse-attention predictions become the sources that caching reuses in later steps. Sparsification error therefore affects both the current sampler update and subsequent reused predictions, making the accuracy of these sources central to the speed–fidelity trade-off.

We study this coupling in direct compositions and make two observations. First, source accuracy matters for fidelity. When we replace the cached tensors with dense counterparts while keeping the reuse schedule and the sparse predictions used by the sampler, the deviation from the dense trajectory decreases. Computing every source densely, however, would forfeit the savings of sparse attention, so we ask whether correction can be shared across steps. Second, we find that sparsification errors remain strongly aligned across nearby denoising steps. This persistence suggests that a single dense correction can serve several sparse evaluations, whose corrected predictions then provide more accurate sources for temporal reuse. Sparse attention and caching should therefore correct each other rather than act independently, as Fig. 2 illustrates. Section 3.1 details both analyses.

Motivated by these observations, we propose RECAP, which connects sparse attention and a two-level cache hierarchy through reciprocal correction, as illustrated in Fig. 3. At the attention level, a cached dense refinement supplies the detail that the current sparse estimate leaves unresolved. As attention inputs evolve across steps, fresh sparse statistics in turn regulate how strongly this aging refinement is applied, so that cached and current information correct each other. At the prediction level, the corrected outputs renew a prediction cache, from which later steps are extrapolated without another transformer evaluation. Sparse computation thus links the two levels, and one dense refresh can support several sparse evaluations together with the prediction reuse that follows them.

This design also naturally supplies the feedback for adaptive online scheduling. Each computed prediction reveals the error of the extrapolation it replaces, and each dense refresh reveals the error of the refinement it renews. Since corrected sparse evaluations renew prediction history while reusing existing refinement, the two caches can be renewed at different rates. The prediction error thus decides when to compute a new prediction, and the refinement error decides whether this computation needs a dense refresh. Under a fixed budget, these decisions choose among prediction reuse, corrected sparse computation, and dense refresh, adapting the schedule to each sample without offline search.

Our contributions are summarized as follows:

- We identify source accuracy and temporal persistence of sparsification error as key opportunities to improve sparse–cache composition. More accurate cache sources recover fidelity, while aligned errors across steps motivate sharing dense correction across sparse evaluations.
- We propose RECAP, which couples sparse attention with hierarchical caching through reciprocal correction. Cached dense refinement corrects sparse attention, corrected predictions renew the sources for whole-step reuse, and online error feedback decides when each cache is renewed.
- We evaluate RECAP on three video diffusion models, where it improves speed–fidelity trade-offs over individual accelerators and direct sparse–cache compositions. On HunyuanVideo, it achieves a 3.05× speedup with higher fidelity than all tested compositions, as Fig. 1 shows.

2 Related Work

Sparse attention. Sparse attention exploits spatial and temporal patterns (Xi et al., 2025; Zhang et al., 2025c; Li et al., 2025), or selects important interactions through content-dependent scores (Xia et al., 2025; Zhang et al., 2025a), token reordering (Yang et al., 2025a; Hu et al., 2026), and query–key clustering (Luo et al., 2026). Training-based approaches further learn sparse patterns (Zhang et al., 2025b; 2026). Instead of discarding omitted interactions, PISA approximates them with block means and shared first-order value compensation (Li et al., 2026), and COBS estimates block mass with

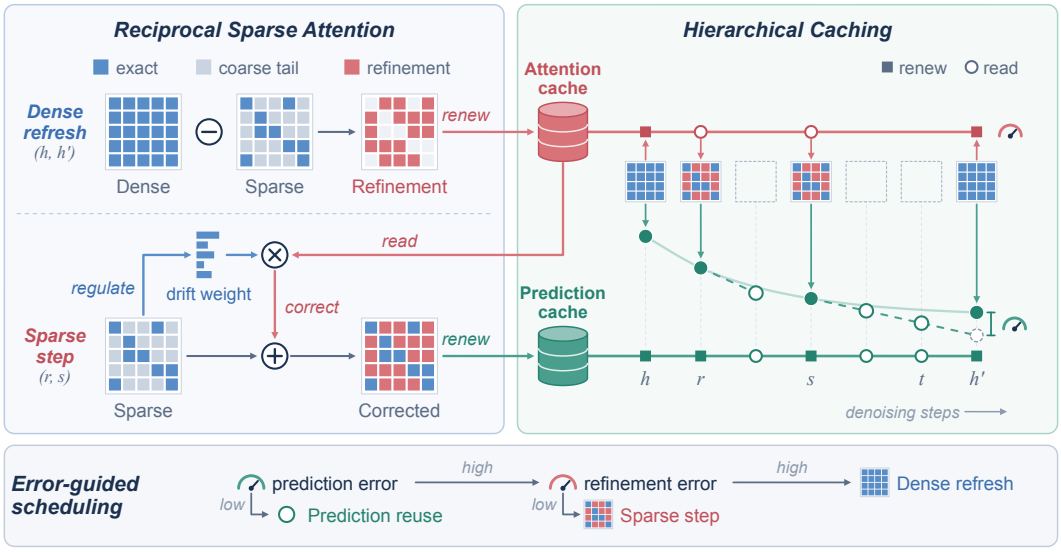


Figure 3: Overview of RECAP. A dense refresh caches the refinement that sparse attention misses, and sparse steps apply it with a drift weight from fresh statistics. Corrected outputs renew the prediction cache used for reuse, and the two errors choose among reuse, a sparse step, and a dense refresh.

second-order key statistics (Tian et al., 2026). Building on these approximations, RECAP corrects the remaining tail value and normalization mass with dense refinement cached from earlier steps.

Temporal caching. Caching exploits similarity across denoising steps by reusing features or outputs (Ma et al., 2024b; Chen et al., 2024; Zhao et al., 2025). LeMiCa optimizes reuse offline (Gao et al., 2025), while other methods adapt reuse to online signals (Liu et al., 2025a; Kahatapitiya et al., 2025; Lv et al., 2026; Ma et al., 2025; Zhou et al., 2025). Further approaches include token-level reuse (Zou et al., 2025), feature extrapolation (Liu et al., 2025b; 2026a), guidance sharing (Lv et al., 2025), and learned routing (Ma et al., 2024a). RECAP uses current sparse statistics to regulate aging refinement and corrected sparse predictions to renew the sources for whole-step prediction reuse. Rather than deciding only when to reuse, RECAP also improves the predictions that are reused.

Combining sparse attention and temporal reuse. DiTFastAttn shares dense-minus-window attention residuals and attention outputs under an offline-selected schedule (Yuan et al., 2024). Re-ttention reuses normalization ratios and output residuals to reconstruct sparse attention (Chen et al., 2025). ASTRAEA uses evolutionary search to optimize per-timestep token budgets (Liu et al., 2026c), while LoSA reuses early-step attention block indices across subsequent denoising steps (Liu et al., 2026b). These methods act in one direction only, whereas RECAP corrects reciprocally and coordinates both cache levels by online error feedback. Appendix Sec. F discusses these distinctions further.

3 METHOD

3.1 MOTIVATION

Background. At denoising step t, a diffusion transformer predicts $p_t = f_\theta(x_t, t, c)$ from latent x_t and conditioning c, and the sampler advances as $x_{t+1} = \Psi_t(x_t, p_t)$. Sparse attention reduces computation within each evaluation by omitting or approximating query-key interactions, while caching reuses earlier features or predictions across steps. These complementary savings make their combination a natural way to further accelerate sampling. However, sparse outputs also become sources for subsequent reuse, so sparsification error can affect both the current sampler update and later steps. This coupling motivates a closer examination of how the two mechanisms interact.

Composition error and source accuracy. We sweep the settings of DFSAttn, TeaCache, and their direct composition in Fig. 4a, finding that combining the two approximations need not cause a disproportionate loss of fidelity. If the two errors simply accumulated, the composed error would match the vector sum of the standalone errors. Instead, it falls below this sum in 103 of 104 configurations in Fig. 4b, revealing a favorable interaction between sparse attention and caching.

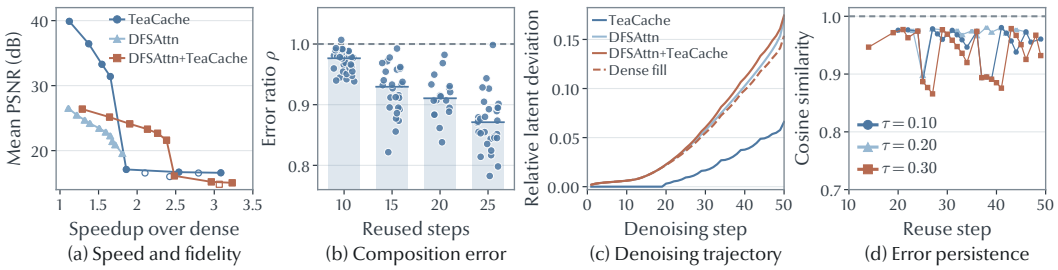


Figure 4: (a) Speed and fidelity of direct composition. (b) Composed error norm relative to the norm of summed standalone errors. (c) Latent deviation with dense cache filling under a fixed reuse schedule. (d) Cosine similarity of sparsification errors between source and reuse steps for different TeaCache thresholds τ. Details of these experiments are provided in Appendix Sec. C.1.

To reduce the remaining error, we focus on source accuracy, since sparse evaluations supply the sources of subsequent reuse. In DFSAttn with TeaCache, we replace cached tensors with dense counterparts computed on the same inputs, while retaining the sparse predictions for sampler updates and the reuse schedule. As Fig. 4c shows, this reduces latent deviation from the matched dense trajectory, demonstrating that *more accurate cache sources can improve the fidelity of the composition.*

Persistence of sparsification error. Computing every source densely, however, would forfeit the savings of sparse attention. We therefore ask whether correction can instead be shared across steps. Along dense trajectories, we compare sparse and dense predictions on identical inputs to isolate sparsification error from trajectory drift. Figure 4d shows that these errors remain strongly aligned across nearby steps for all thresholds, so a correction measured at the source step still points in the right direction at later steps. This suggests that *a single dense correction can support multiple sparse evaluations* by combining a fresh sparse estimate with cached detail that is costly to recompute.

Method overview. RECAP connects these insights through a two-level cache hierarchy, as illustrated in Fig. 3. The persistence of sparsification error motivates sharing dense correction across sparse evaluations, which we develop as attention-level refinement in Sec. 3.2. The resulting improved predictions then serve as more accurate sources for whole-step temporal reuse in Sec. 3.3. Finally, Sec. 3.4 coordinates renewal at both levels through online error feedback. At each step, RECAP either reuses an extrapolated prediction, performs corrected sparse computation, or refreshes densely.

3.2 ATTENTION-LEVEL CACHING FOR SPARSE RECONSTRUCTION

The source-accuracy findings in Sec. 3.1 motivate improving sparse predictions before reuse, and the temporal alignment of sparsification error suggests sharing correction across steps. Since this error originates in the omitted interactions, we realize both ideas at the attention level, where a fresh coarse estimate tracks changes in omitted attention and cached dense refinement supplies the discrepancy that this estimate leaves unresolved. Both are maintained separately for each layer and attention head.

Selected interactions and coarse tail. For scaled dot-product attention (Vaswani et al., 2017), we partition the keys for a query q into a selected set H and a tail T omitted from exact computation. We select key blocks by pooled query-key scores and compute attention over H exactly. The omitted tail contributes both attention values and normalization mass, so reconstructing the full output requires estimating both. For either set A, let o_A be its normalized attention output and $\ell_A = \log \sum_{j \in A} \exp(\beta q^\top k_j)$ its log-mass, where $\beta = d_k^{-1/2}$. The full output is

$$o_t = \frac{e^{\ell_{H,t}} o_{H,t} + e^{\ell_{T,t}} o_{T,t}}{e^{\ell_{H,t}} + e^{\ell_{T,t}}}. \tag{1}$$

For the omitted tail, block means provide initial estimates of value and normalization mass, which we refine through two moment corrections. Key variance refines the log-mass through the second-order approximation of COBS (Tian et al., 2026), while pooled value-key covariance refines the value through the first-order compensation of PISA (Li et al., 2026). The logarithm of the summed block masses gives the coarse log-mass $\bar{\ell}_{T,t}$, while weighting block values by their normalized masses gives the coarse output $\bar{o}_{T,t}$. These statistics are recomputed at each sparse evaluation, allowing the estimate to track attention changes without evaluating every omitted query-key interaction.

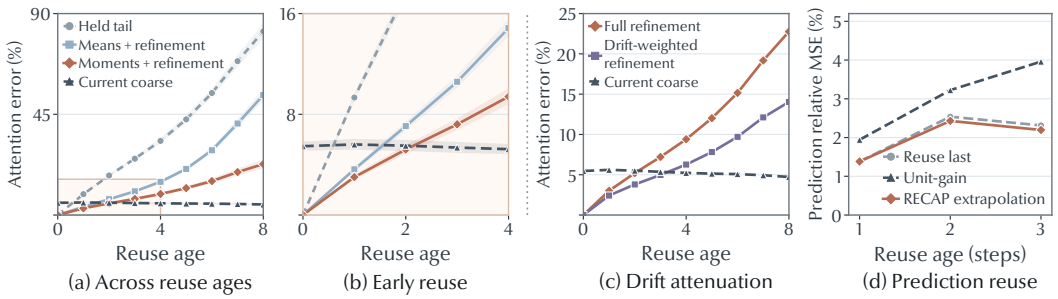


Figure 5: Attention refinement and prediction reuse. (a, b) Reconstruction error growth with reuse age. (c) Drift weighting moderates error growth relative to full-strength refinement. (d) Fitted gain extrapolation compared with holding the latest prediction and unit-gain extrapolation.

Cached dense refinement. The coarse estimate leaves a discrepancy that a dense refresh can recover. At refresh h, we subtract the selected branch's mass and unnormalized value contribution from those of full attention to obtain the exact tail output $o_{T,h}$ and log-mass $\ell_{T,h}$. We cache their differences from the coarse estimates and add these residuals to fresh estimates at a later sparse evaluation t,

$$\begin{aligned} d_h^v &= o_{T,h} - \bar{o}_{T,h}, \qquad d_h^\ell = \ell_{T,h} - \bar{\ell}_{T,h}, \\ \widehat{o}_{T,t} &= \bar{o}_{T,t} + d_h^v, \qquad \widehat{\ell}_{T,t} = \bar{\ell}_{T,t} + d_h^\ell. \end{aligned} \tag{2}$$

Each layer retains its selected partition between dense refreshes, so that cached residuals and the current coarse estimates continue to describe the same set of omitted keys. The partition can then be re-selected at each dense refresh, together with the cached residuals, as attention evolves.

To see why this decomposition helps, write either tail quantity as $y_t = \bar{y}_t + d_t$, so that the reconstruction error becomes $\widehat{y}_t - y_t = d_h - d_t$. The fresh estimate absorbs changes in the coarse component, leaving reconstruction error determined only by changes in the unresolved discrepancy. As Fig. 5a and Fig. 5b illustrate, updating the coarse component slows error growth relative to holding the entire tail fixed, while adding cached refinement further improves accuracy at early reuse ages.

Drift-weighted refinement. However, Fig. 5b also shows that the benefit of cached refinement diminishes with reuse age as attention inputs evolve. We therefore use the fresh coarse statistics to regulate the cached contribution, so that current sparse computation in turn corrects the cache when earlier refinement becomes less reliable. To quantify this drift, each dense refresh also stores the coarse log-mass $\bar{\ell}_{T,i,h}$ alongside the residuals. For video query row i, the change $u_{i,t} = \bar{\ell}_{T,i,t} - \bar{\ell}_{T,i,h}$ captures how far current tail statistics have moved from those at the refresh. We center this over the N_q video query rows within each sample and attention head to obtain the refinement weight

$$\Delta_{i,t} = u_{i,t} - \frac{1}{N_q}\sum_{j=1}^{N_q} u_{j,t}, \qquad a_{i,t} = e^{-|\Delta_{i,t}|}. \tag{3}$$

Centering removes the common log-mass shift already carried by the fresh estimate. The weight thus decreases only with row-specific deviations and introduces no tunable parameter. To keep value and mass consistent, the same weight blends their unnormalized contributions,

$$\begin{aligned} \widetilde{Z}_{T,t} &= (1 - a_t)e^{\bar{\ell}_{T,t}} + a_t e^{\widehat{\ell}_{T,t}}, \\ \widetilde{N}_{T,t} &= (1 - a_t)e^{\bar{\ell}_{T,t}}\bar{o}_{T,t} + a_t e^{\widehat{\ell}_{T,t}}\widehat{o}_{T,t}. \end{aligned} \tag{4}$$

The normalized output $\widetilde{o}_{T,t} = \widetilde{N}_{T,t}/\widetilde{Z}_{T,t}$ and log-mass $\widetilde{\ell}_{T,t} = \log \widetilde{Z}_{T,t}$ then merge with the selected branch through Eq. (1). As shown in Fig. 5c, this weighting moderates reconstruction error growth compared with applying refinement at full strength as the cached correction ages. Appendix Sec. A provides the mathematical details of the moment estimates, tail recovery, and blending.

3.3 Prediction-Level Caching for Step Reuse

Since more accurate sources improve the fidelity of temporal reuse, the corrected sparse predictions from Sec. 3.2 naturally serve as better anchors. This motivates caching and extrapolating these improved outputs to advance subsequent sampler steps without another transformer evaluation.

Prediction extrapolation. To account for changes between steps, we estimate a local rate of change from the two most recent computed predictions at steps $r < s$ and extrapolate to a later step t,

$$v_s = \frac{p_s - p_r}{s - r}, \qquad \widehat{p}_t = p_s + \alpha_s(t - s)v_s, \tag{5}$$

where intervals count sampler steps and the scalar gain α_s controls how strongly the observed change is carried forward. Because prediction changes may not continue at the same rate, we fit the gain online. At each computed step t with two preceding sources, we compare the observed change $y_t = p_t - p_s$ with the unit-gain extrapolation $z_t = (t - s)v_s$ through a cumulative least-squares fit,

$$\alpha_t = \arg\min_{a \in \mathbb{R}} \sum_{j \in \mathcal{C}_t} \|y_j - a z_j\|_F^2, \tag{6}$$

where $\mathcal{C}_t$ contains computed steps through t with two preceding sources. During prediction reuse, the sampler advances with $\widehat{p}_t$ while the transformer evaluation is skipped and both the prediction history and gain remain fixed. As Fig. 5d illustrates, the fitted gain reduces extrapolation error relative to unit gain and improves over simply holding the latest prediction at longer reuse ages. This comparison fixes prediction histories and latent inputs to isolate the effect of the extrapolation rule.

Prediction-source renewal. Each computed prediction updates both the anchor and the rate of change, so source accuracy directly affects both terms in Eq. (5). Corrected sparse evaluations renew these quantities while continuing to reuse existing attention refinement, allowing prediction history to advance between dense refreshes without requiring a new dense correction. A single dense refresh can therefore support several sparse evaluations and their subsequent prediction reuse. Restricting this renewal to dense refreshes substantially reduces fidelity, as Sec. 4.3 shows. Appendix Sec. B.1 derives the closed-form gain and shows how source errors are carried into reused predictions.

3.4 Error-Guided Online Scheduling

The two levels of reuse in RECAP naturally supply the feedback needed to schedule them. Each computed prediction reveals the error of the extrapolation it replaces, and each dense refresh reveals the error of the refinement it renews. RECAP uses these errors to decide online when to compute a new prediction and whether this computation needs a dense refresh. The schedule therefore adapts to each sample as sampling proceeds, rather than relying on offline search or a fixed reuse pattern.

Prediction feedback decides when to compute. We compare each computed prediction p_t with its extrapolation $\widehat{p}_t$, normalizing by the elapsed steps since extrapolation error grows with the interval,

$$e_{\mathrm{pred},t} = \frac{\|p_t - \widehat{p}_t\|_F}{\|p_t\|_F}, \qquad \rho_{\mathrm{pred},t} = \frac{e_{\mathrm{pred},t}}{t - s}, \tag{7}$$

where s is the latest source step. A rate below τ_{pred} extends the next reuse interval, which is capped by spreading the remaining evaluations of a budget B evenly over the remaining steps. This cap shortens when the rate exceeds its running mean, which concentrates computation where predictions change quickly. The budget thus fixes how much is computed, while the feedback decides where.

Refinement feedback decides how to compute. Corrected sparse attention is used only while the cached refinement remains reliable, which requires a dense reference to verify. At each dense refresh h, we compare the dense attention output $O^D_{l,h}$ of each layer l with the corrected sparse output $\widetilde{O}^{\mathrm{old}}_{l,h}$ reconstructed from the old refinement. Pooling the layer errors before the residual update gives

$$e_{\mathrm{ref},h} = \left(\frac{\sum_l \|\widetilde{O}^{\mathrm{old}}_{l,h} - O^D_{l,h}\|_F^2}{\sum_l \|O^D_{l,h}\|_F^2} \right)^{1/2}. \tag{8}$$

Since sparsification error remains aligned across nearby steps, as Fig. 4d shows, this error indicates how long the new refinement can be trusted. A passing check extends its permitted age, whereas a failed check makes the next computed prediction dense. Between checks, drift weighting adjusts the cached contribution per query row, while the check decides when to renew the whole refinement.

Together, the two decisions yield the three actions in Fig. 3, namely prediction reuse, corrected sparse computation that renews only the prediction level, and a dense refresh that renews both cache levels. Each computed step thus supplies both a new source for reuse and the feedback that decides when the two caches are renewed. Appendix Sec. B gives the update rules and sampling algorithm.

Table 1: Main results on CogVideoX-1.5 and HunyuanVideo. Re-ttention exceeds GPU memory on HunyuanVideo and is therefore omitted. VBench Overall averages the six evaluated dimensions after per-dimension normalization. Best and second-best results are bold and underlined, respectively.

Method	Fidelity			Quality					Efficiency	
	PSNR↑	SSIM↑	LPIPS↓	CLIPIQA+↑	IQ↑	AQ↑	TF↑	Overall↑	Latency↓	Speedup↑
CogVideoX-1.5 (768×1360, 81 frames)										
Dense	–	–	–	0.489	64.05	61.33	97.06	83.31	1,106s	1.00×
TeaCache (CVPR'25)	31.13	0.9472	0.0475	0.472	63.16	60.95	97.06	83.06	664s	1.67×
SVG (ICML'25)	23.17	0.8044	0.1952	0.478	63.41	60.16	96.76	82.69	776s	1.43×
Re-ttention (NeurIPS'25)	22.42	0.7587	0.2557	0.373	57.67	56.38	96.98	81.16	837s	1.32×
LeMiCa (NeurIPS'25)	32.06	0.9631	0.0359	0.480	63.31	61.05	97.07	83.09	700s	1.58×
SVOO (ICML'26)	25.88	0.8532	0.1484	0.451	63.50	60.12	96.91	82.75	778s	1.42×
DFSAttn (ICML'26)	26.13	0.8694	0.1266	0.480	63.32	60.58	96.88	82.85	685s	1.61×
NaviCache (ICML'26)	26.99	0.8788	0.1123	0.468	63.22	60.68	**97.10**	83.04	654s	1.69×
RECAP-SPEED	33.69	0.9675	0.0306	0.483	63.56	61.08	97.09	83.16	**632s**	**1.75×**
RECAP-QUALITY	**35.80**	**0.9766**	**0.0223**	**0.486**	**63.79**	**61.28**	97.06	**83.23**	681s	1.62×
HunyuanVideo (720×1280, 129 frames)										
Dense	–	–	–	0.512	64.91	58.74	98.92	85.13	3,440s	1.00×
TeaCache (CVPR'25)	21.94	0.8376	0.2073	0.495	62.82	57.96	98.92	84.57	1,638s	2.10×
SVG (ICML'25)	21.13	0.8203	0.2458	0.490	63.76	58.26	98.76	84.51	1,641s	2.10×
LeMiCa (NeurIPS'25)	35.12	0.9730	0.0410	**0.513**	63.12	58.45	99.03	84.87	1,746s	1.97×
SVOO (ICML'26)	27.68	0.9048	0.1218	0.505	64.11	58.40	98.91	84.85	1,831s	1.88×
DFSAttn (ICML'26)	30.67	0.9420	0.0624	0.504	64.14	58.45	98.80	84.78	1,697s	2.03×
NaviCache (ICML'26)	35.29	0.9720	0.0310	0.502	**64.63**	58.68	98.94	85.02	1,761s	1.95×
RECAP-SPEED	35.35	0.9744	0.0268	0.508	63.82	58.66	**99.07**	85.04	**1,611s**	**2.14×**
RECAP-QUALITY	**36.71**	**0.9795**	**0.0189**	**0.513**	64.27	**58.73**	98.96	**85.08**	1,727s	1.99×

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models and baselines. We evaluate RECAP on CogVideoX-1.5-5B (Yang et al., 2025b), HunyuanVideo-13B (Kong et al., 2024), and Wan2.1-1.3B (Team Wan et al., 2025), each with 50 steps. Wan2.1 results are in Appendix Sec. E.1. Baselines include SVG (Xi et al., 2025), Re-ttention (Chen et al., 2025), SVOO (Luo et al., 2026), and DFSAttn (Hu et al., 2026) for sparse attention and TeaCache (Liu et al., 2025a), LeMiCa (Gao et al., 2025), and NaviCache (Lv et al., 2026) for caching. Pairing SVOO and DFSAttn with each caching baseline yields six direct compositions. Appendix Sec. D extends RECAP to FLUX.1-dev (Black Forest Labs, 2024) image generation.

Evaluation metrics. We use optimized VBench prompts (Huang et al., 2024), with initial noise, guidance, and sampler settings matched across methods within each comparison. Fidelity to dense generation is measured using PSNR (Horé & Ziou, 2010), SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) on paired lossless frames. CLIP-IQA+ (Wang et al., 2023) and VBench (Huang et al., 2024) assess generation quality. For VBench, we average six quality dimensions after its official per-dimension normalization, as detailed in Appendix Sec. E.2. The main tables also report the individual imaging quality (IQ), aesthetic quality (AQ), and temporal flickering (TF) scores.

Implementation details. The main experiments on CogVideoX-1.5 and Wan2.1-1.3B are conducted on NVIDIA RTX 4090 GPUs, while those on HunyuanVideo use NVIDIA A800 GPUs. We use FlashAttention-2 (Dao, 2024) for dense refreshes, the Block Sparse Attention library (Guo et al., 2024) for selected interactions, and custom Triton kernels (Tillet et al., 2019) for coarse-tail estimation and refinement. RECAP requires no training or model fine-tuning. We select its computation budget and error tolerances using the lightweight parameter selection procedure in Appendix Sec. C.4. Appendix Sec. C provides the full parameter settings for each model and describes the memory optimizations.

4.2 COMPARISON WITH EXISTING METHODS

Quantitative results. Table 1 compares individual methods in the normal-speedup regime. RECAP-QUALITY improves PSNR over the strongest individual baseline by 3.74 dB on CogVideoX-1.5 and 1.42 dB on HunyuanVideo. RECAP-SPEED achieves the lowest latency in this comparison and reaches speedups of 1.75× and 2.14× over dense generation, respectively. Both settings outperform

Table 2: Comparison with direct sparse–cache compositions, each configured to a similar speedup in the high-speed regime. Best and second-best results are bold and underlined, respectively.

Method	Fidelity			Quality					Efficiency	
	PSNR↑	SSIM↑	LPIPS↓	CLIPIQA+↑	IQ↑	AQ↑	TF↑	Overall↑	Latency↓	Speedup↑
CogVideoX-1.5 (768×1360, 81 frames)										
DFSAttn + TeaCache	15.89	0.5654	0.4245	0.422	61.59	58.32	95.93	81.55	446s	2.48×
DFSAttn + LeMiCa	21.57	0.7835	0.1934	0.437	63.43	60.76	95.91	82.30	440s	2.51×
DFSAttn + NaviCache	21.42	0.7648	0.2059	0.434	63.39	60.42	96.04	82.23	434s	2.54×
SVOO + TeaCache	15.77	0.5594	0.4206	0.419	62.79	56.69	95.83	81.41	439s	2.52×
SVOO + LeMiCa	21.14	0.7627	0.2043	0.430	63.34	60.34	96.07	82.32	438s	2.52×
SVOO + NaviCache	20.43	0.7047	0.2816	0.405	62.95	58.34	96.17	81.72	427s	2.59×
RECAP	**25.35**	**0.8724**	**0.1421**	**0.439**	**63.61**	**60.86**	**96.30**	**82.64**	**415s**	**2.67×**
HunyuanVideo (720×1280, 129 frames)										
DFSAttn + TeaCache	24.30	0.8701	0.1545	0.464	**62.71**	58.14	98.43	83.91	1,213s	2.83×
DFSAttn + LeMiCa	28.49	0.9146	0.0905	0.470	60.69	57.80	98.69	83.79	1,283s	2.68×
DFSAttn + NaviCache	28.46	0.9148	0.0875	0.464	62.18	58.14	98.48	83.80	1,350s	2.55×
SVOO + TeaCache	21.27	0.8092	0.2535	0.461	60.95	57.85	98.45	83.52	1,447s	2.38×
SVOO + LeMiCa	27.39	0.8999	0.1266	**0.471**	61.18	57.97	98.73	83.86	1,329s	2.59×
SVOO + NaviCache	27.16	0.8976	0.1280	0.467	62.31	58.35	98.64	83.97	1,560s	2.21×
RECAP	**29.67**	**0.9392**	**0.0731**	0.467	62.35	**58.68**	**98.81**	**84.34**	**1,129s**	**3.05×**

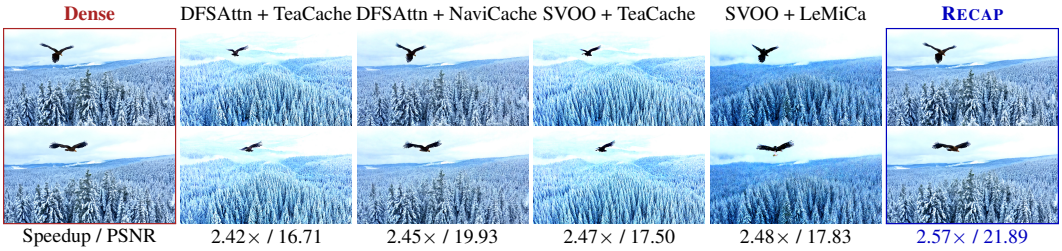


Figure 6: Samples from CogVideoX-1.5 (top), HunyuanVideo (middle), and FLUX.1-dev (bottom). More image generation results are provided in Appendix Sec. D. Best viewed zoomed in.

all individual baselines in fidelity and VBench Overall on both models. In the high-speed regime, Tab. 2 compares direct sparse–cache compositions. RECAP exceeds the highest-PSNR composition, DFSAttn with LeMiCa, by 3.78 dB and 1.18 dB on the two models, respectively. It also achieves the best SSIM, LPIPS, and VBench Overall scores alongside the lowest latency, with speedups of 2.67× and 3.05×. On CogVideoX-1.5, RECAP achieves the best result on every reported metric.

Qualitative results. The samples in Fig. 6 show that RECAP agrees more closely with dense generation in both fine details and overall appearance. On CogVideoX-1.5, RECAP better preserves the eagle's posture, the details of the snow-covered forest, and the overall color of the scene, whereas

Table 3: Ablation studies in the normal-speedup regime, with fast-regime results in Tab. 10. Latency is measured on one NVIDIA A800 GPU. Bold marks the best value within each comparison.

Variant	PSNR↑	SSIM↑	LPIPS↓
W/o refinement	34.57	0.9749	0.0303
Frozen state	30.12	0.9396	0.0747
RECAP	**36.15**	**0.9817**	**0.0219**

(a) Reconstruction components.

	Same locations			*Matched runtime*		
Variant	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PISA	29.55	0.9351	0.0755	31.84	0.9549	0.0532
Re-ttention	26.93	0.8677	0.1329	28.68	0.8983	0.1036
RECAP	**36.15**	**0.9817**	**0.0219**	**36.15**	**0.9817**	**0.0219**

(b) Alternative reconstructions.

Variant	PSNR↑	SSIM↑	LPIPS↓
Dense-only history renewal	31.80	0.9623	0.0468
Uniform schedule	35.70	0.9796	0.0254
RECAP	**36.15**	**0.9817**	**0.0219**

(c) Prediction-history renewal and online scheduling.

Variant	PSNR↑	SSIM↑	LPIPS↓	Latency
W/o prediction reuse	38.79	0.9878	0.0143	845.51s
RECAP (matched latency)	**41.45**	**0.9934**	**0.0064**	843.97s
RECAP	36.15	0.9817	0.0219	**619.69s**

(d) Prediction reuse.

three of the four compositions shift it toward a saturated blue. On HunyuanVideo, it more faithfully reproduces the details of the cat's face and the water, and renders the fur texture more clearly. On FLUX.1-dev, it is the only method that preserves both the rectangular window and the eight children. Additional comparisons across models and speedup regimes are provided in Appendix Sec. E.3.

4.3 Ablation Studies

We ablate each component of RECAP on CogVideoX-1.5, namely attention reconstruction, prediction reuse, and online scheduling. Each variant changes one component with all other settings fixed. Appendix Sec. E.4 repeats every ablation at the fast setting of Tab. 2, with the same conclusions.

Attention reconstruction. The reconstruction in Sec. 3.2 pairs fresh coarse estimates with cached dense refinement. Removing the refinement at the same computation locations lowers PSNR from 36.15 to 34.57 dB in Tab. 3a. Freezing the initial refinement and partition lowers it further to 30.12 dB. The refinement thus supplies detail missing from the coarse estimate and must be renewed during sampling. Adapted PISA (Li et al., 2026) and Re-ttention (Chen et al., 2025) change the cost of each sparse evaluation. Table 3b therefore compares them with our reconstruction both at the same locations and at matched runtime. RECAP leads in both, by 4.31 and 7.47 dB at matched runtime.

Prediction reuse. Section 3.3 extrapolates reused steps from a prediction history that corrected sparse evaluations also renew. Restricting this renewal to dense refreshes lowers PSNR from 36.15 to 31.80 dB in Tab. 3c, showing that corrected sparse evaluations are valuable reuse sources. Disabling reuse and computing all 50 steps raises PSNR to 38.79 dB, but latency grows from 619.69 to 845.51 s, so reuse saves 27% of runtime. At matched latency, RECAP instead reaches 41.45 dB in Tab. 3d.

Online scheduling. The scheduler in Sec. 3.4 places computed steps by prediction error, concentrating them where predictions change quickly. Spacing the same number of steps evenly lowers PSNR from 36.15 to 35.70 dB in Tab. 3c. Error-guided placement thus improves fidelity at no extra cost.

5 Conclusion

We presented RECAP, a training-free method that couples sparse attention and hierarchical caching through reciprocal correction instead of composing them directly. We showed that the accuracy of cache sources affects the fidelity of the composition and that sparsification errors persist across nearby steps. Building on these findings, RECAP maintains a two-level cache hierarchy. At the attention level, cached dense refinement corrects sparse attention, while fresh sparse statistics regulate how strongly this aging refinement is applied. At the prediction level, corrected predictions renew the sources for whole-step reuse. Online error feedback decides when to compute and when to refresh densely, so the schedule adapts to each sample without costly offline search. Across three video diffusion models, RECAP improves the speed–fidelity trade-off over individual accelerators and direct compositions, reaching a $3.05\times$ speedup on HunyuanVideo with higher fidelity than all tested compositions.

AI USE STATEMENT

In this work, we used generative AI tools for polishing language to improve clarity. We have not used generative AI tools to generate research ideas, conduct the reported experiments, or generate experimental results. We have reviewed all AI-assisted work, checking revisions for technical accuracy and consistency with our method and experimental evidence. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This research adheres to the ICLR Code of Ethics. It uses publicly available video-generation models and focuses on algorithmic improvements for inference acceleration. We identify no ethical concerns specific to the proposed acceleration method.

REPRODUCIBILITY STATEMENT

Method details are provided in Sec. 3, Appendix Sec. A, and Appendix Sec. B. Experimental configurations and evaluation metrics are described in Sec. 4.1 and Appendix Sec. C. Video samples are provided in the supplementary materials. We will release the code after the paper is accepted.

REFERENCES

Black Forest Labs. FLUX.1 [dev]. https://huggingface.co/black-forest-labs/FLUX.1-dev, 2024. 7, 20

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Δ-DiT: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024. 3

Ruichen Chen, Keith G. Mills, Liyao Jiang, Chao Gao, and Di Niu. Re-ttention: Ultra sparse visual generation via attention statistical reshape. In *NeurIPS*, 2025. 3, 7, 9, 26

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024. 7

Huanlin Gao, Ping Chen, Fuyuan Shi, Chao Tan, Zhaoxiang Liu, Fang Zhao, Kai Wang, and Shiguo Lian. LeMiCa: Lexicographic minimax path caching for efficient diffusion-based video generation. In *NeurIPS*, 2025. 3, 7, 20, 26

Junxian Guo, Haotian Tang, Shang Yang, Zhekai Zhang, Zhijian Liu, and Song Han. Block Sparse Attention. https://github.com/mit-han-lab/Block-Sparse-Attention, 2024. 7

Jiaqi Han, Juntong Shi, Puheng Li, Haotian Ye, Qiushan Guo, and Stefano Ermon. Adaptive spectral feature forecasting for diffusion sampling acceleration. In *CVPR*, 2026. 20

Alain Horé and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *ICPR*, 2010. 7, 20

Jie Hu, Zixiang Gao, Yutong He, and Kun Yuan. DFSAttn: Dynamic fine-grained sparse attention for efficient video generation. In *ICML*, 2026. 1, 2, 7

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024. 7

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S. Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. In *ICCV*, 2025. 3

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 1, 7

Haopeng Li, Shitong Shao, Wenliang Zhong, Zikai Zhou, Lichen Bai, Hui Xiong, and Zeke Xie. PISA: Piecewise sparse attention is wiser for efficient diffusion transformers. *arXiv preprint arXiv:2602.01077*, 2026. 2, 4, 9, 14, 26

Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial attention: $O(n \log n)$ sparse attention with energy decay for long video generation. In *NeurIPS*, 2025. 2

Enhuai Liu, Yunke Wang, Changming Sun, and Chang Xu. D2Cache: Second-order delta caching for higher video diffusion acceleration. In *CVPR*, 2026a. 3

Enhuai Liu, Yunke Wang, Yutong Wang, Changming Sun, and Chang Xu. LoSA: Near-lossless sparse attention for training-free video diffusion acceleration. *arXiv preprint arXiv:2608.12032*, 2026b. 3, 26

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *CVPR*, 2025a. 1, 3, 7, 26

Haosong Liu, Yuge Cheng, Wenxuan Miao, Zihan Liu, Aiyue Chen, Jing Lin, Yiwu Yao, Chen Chen, Jingwen Leng, Minyi Guo, and Yu Feng. Astraea: A token-wise acceleration framework for video diffusion transformers. In *ICLR*, 2026c. 3, 26

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *ICCV*, 2025b. 3

Jiayi Luo, Jiayu Chen, Jiankun Wang, Cong Wang, Hanxin Zhu, Qingyun Sun, Chen Gao, Zhibo Chen, and Jianxin Li. Attention sparsity is input-stable: Training-free sparse attention for video generation via offline sparsity profiling and online QK co-clustering. In *ICML*, 2026. 2, 7

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. FasterCache: Training-free video diffusion model acceleration with high quality. In *ICLR*, 2025. 3

Zheqi Lv, Zhibo Zhu, Jinke Wang, Qi Tian, Shengyu Zhang, Zhengyu Chen, Chengxi Zang, Zhou Zhao, and Fei Wu. NaviCache: Test-time self-calibration caching for video generation. In *ICML*, 2026. 3, 7

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-Cache: Accelerating diffusion transformer via layer caching. In *NeurIPS*, 2024a. 3

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *CVPR*, 2024b. 3

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. MagCache: Fast video generation with magnitude-aware cache. In *NeurIPS*, 2025. 3

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 1

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Raphael Gontijo-Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 20

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 7

Alexander Tian, Aditya Ghai, Sanjit Neelam, Zaal Vasania, and Akshay Mishra. COBS: Cumulant order block sparse attention. *arXiv preprint arXiv:2607.09052*, 2026. 3, 4, 14, 26

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *MAPL*, 2019. 7

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

Jianyi Wang, Kelvin C. K. Chan, and Chen Change Loy. Exploring CLIP for assessing the look and feel of images. In *AAAI*, 2023. 7

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *TIP*, 2004. 7, 20

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse Video-Gen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *ICML*, 2025. 1, 2, 7

Yifei Xia, Suhan Ling, Fangcheng Fu, Yujie Wang, Huixia Li, Xuefeng Xiao, and Bin Cui. Training-free and adaptive sparse attention for efficient long video generation. In *ICCV*, 2025. 2

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *NeurIPS*, 2023. 20

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse VideoGen2: Accelerate video generation with sparse attention via semantic-aware permutation. In *NeurIPS*, 2025a. 2

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-video diffusion models with an expert transformer. In *ICLR*, 2025b. 1, 7

Zhihang Yuan, Hanling Zhang, Pu Lu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. DiTFastAttn: Attention compression for diffusion transformer models. In *NeurIPS*, 2024. 3, 26

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *ICML*, 2025a. 2

Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, Joseph E. Gonzalez, Jianfei Chen, and Jun Zhu. SLA: Beyond sparsity in diffusion transformers via fine-tunable sparse–linear attention. In *ICLR*, 2026. 2

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric P. Xing, and Hao Zhang. Faster video diffusion with trainable sparse attention. In *NeurIPS*, 2025b. 2

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *ICML*, 2025c. 2

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7, 20

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In *ICLR*, 2025. 3

Xin Zhou, Dingkang Liang, Kaijin Chen, Tianrui Feng, Xiwu Chen, Hongkai Lin, Yikang Ding, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Less is enough: Training-free video diffusion acceleration via runtime-adaptive caching. *arXiv preprint arXiv:2507.02860*, 2025. 3

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *ICLR*, 2025. 3

TABLE OF CONTENTS

A Attention Reconstruction . . . 14
A.1 Current Tail Estimation . . . 14
A.2 Dense Refinement and Reconstruction Error . . . 15
B Prediction Reuse and Online Scheduling . . . 16
B.1 Online Prediction Update . . . 16
B.2 Error-Guided Scheduling . . . 17
C Implementation Details . . . 18
C.1 Motivation Experiments . . . 18
C.2 Attention Computation . . . 18
C.3 Memory Optimization . . . 19
C.4 Parameter Selection . . . 19
D Image Generation . . . 20
E Additional Results . . . 21
E.1 Results on Wan2.1-1.3B . . . 21
E.2 Detailed VBench Results . . . 21
E.3 More Visual Results . . . 23
E.4 More Ablations . . . 25
F Extended Related Work . . . 26
G Limitations . . . 26

A ATTENTION RECONSTRUCTION

This section develops the reconstruction in Sec. 3.2. We first derive the current tail estimate from block moments, then show how dense attention supplies its cached residual. Finally, we derive how drift weighting affects the tail value, normalization mass, and reconstructed attention output. We suppress the layer, attention-head, and timestep indices when considering a single attention evaluation.

A.1 CURRENT TAIL ESTIMATION

Block moments. For a block b containing n_b tokens, let $\bar{k}_b$ and $\bar{v}_b$ denote its mean key and value. Write the centered vectors as $\delta k_j = k_j - \bar{k}_b$ and $\delta v_j = v_j - \bar{v}_b$, and define the key covariance and value-key covariance by

$$C_b = \frac{1}{n_b} \sum_{j \in b} \delta k_j \delta k_j^\top, \qquad C_b^{vk} = \frac{1}{n_b} \sum_{j \in b} \delta v_j \delta k_j^\top. \tag{9}$$

These moments describe the variation that block means leave unresolved. For a query q, the centered logit is $z_j = \beta q^\top \delta k_j$, where $\beta = d_k^{-1/2}$. Factoring the mean logit out of the block's normalization mass gives

$$\ell_b = \log n_b + \beta q^\top \bar{k}_b + K_b(1), \qquad K_b(a) = \log \left(n_b^{-1} \sum_{j \in b} e^{a z_j} \right). \tag{10}$$

Because the logits are centered, $K_b(0) = K_b'(0) = 0$ and $K_b''(0) = \beta^2 q^\top C_b q$. Retaining the second-order term therefore gives the log-mass correction $\frac{1}{2}\beta^2 q^\top C_b q$, following the moment approximation of COBS (Tian et al., 2026).

The value correction follows from the normalized output $o_b(a) = \sum_{j \in b} e^{a z_j} v_j / \sum_{j \in b} e^{a z_j}$. At $a = 0$, all tokens in the block receive equal weight, giving $o_b(0) = \bar{v}_b$. Differentiating at zero gives $o_b'(0) = \beta C_b^{vk} q$, so the first-order approximation is $o_b(1) \approx \bar{v}_b + \beta C_b^{vk} q$. This is the value compensation underlying PISA (Li et al., 2026).

Shared value correction and tail aggregation. We use a shared value-key covariance to capture the average variation within blocks. For the set $\mathcal{B}$ of video key blocks, define

$$N = \sum_{b\in\mathcal{B}} n_b, \qquad M = \frac{1}{N}\sum_{b\in\mathcal{B}} n_b C_b^{vk}, \qquad \delta v(q) = \beta M q. \tag{11}$$

This correction is common to all blocks for a given query, while the key covariance supplies a distinct mass correction for each block. The resulting basic moment estimates are

$$L_b = \log n_b + \beta q^\top \bar{k}_b + \tfrac{1}{2}\beta^2 q^\top C_b q, \qquad \bar{v}_b^{\mathrm{corr}} = \bar{v}_b + \delta v(q). \tag{12}$$

The covariance representation and treatment of large logit variation used in their numerical evaluation are specified in Appendix Sec. C.2. For the omitted blocks $\mathcal{T} \subseteq \mathcal{B}$, their estimated masses determine both the tail normalization and the weights for combining their values,

$$\begin{aligned} \bar{\ell}_T &= \log \sum_{b\in\mathcal{T}} e^{L_b}, \quad \pi_b = e^{L_b - \bar{\ell}_T}, \\ \bar{o}_T &= \sum_{b\in\mathcal{T}} \pi_b \bar{v}_b^{\mathrm{corr}}. \end{aligned} \tag{13}$$

Thus, the coarse estimate represents the omitted tail through current block statistics. Recomputing these statistics allows it to follow changing inputs while the remaining discrepancy is supplied by cached dense refinement.

A.2 Dense Refinement and Reconstruction Error

Dense tail recovery. At a dense refresh, the full output o and log-mass ℓ determine the normalization mass $Z = e^{\ell}$ and unnormalized value sum Zo. The selected branch similarly contributes $Z_H = e^{\ell_H}$ and $Z_H o_H$. Subtracting these contributions leaves $Z_T = Z - Z_H$ and $Z_T o_T = Zo - Z_H o_H$ for the omitted tail. Writing $r = Z_H/Z$ gives

$$r = \exp(\ell_H - \ell), \qquad o_T = \frac{o - r o_H}{1 - r}, \qquad \ell_T = \ell + \log(1 - r). \tag{14}$$

Subtracting the coarse estimates from these exact tail quantities produces the residuals in Eq. (2). Keeping the selected partition fixed between dense refreshes ensures that the residuals and subsequent coarse estimates refer to the same omitted keys.

Reconstruction error. For either the tail output or log-mass, write the exact quantity as $y_t = \bar{y}_t + d_t$. Reusing the residual from refresh h gives $\widehat{y}_t = \bar{y}_t + d_h$, whereas holding the entire earlier tail gives $y_h = \bar{y}_h + d_h$. Their errors are

$$\widehat{y}_t - y_t = d_h - d_t, \qquad y_h - y_t = \bar{y}_h - \bar{y}_t + d_h - d_t. \tag{15}$$

Updating the coarse estimate removes the term $\bar{y}_h - \bar{y}_t$ associated with changes in the block statistics. The remaining error depends on how the unresolved discrepancy has changed since the dense refresh, which explains the role of sharing residual correction while continuing to compute the coarse component.

Mass-aware blending. The weight a_t in Eq. (3) blends the fresh coarse estimate with the estimate augmented by cached refinement. Substituting Eq. (2) into Eq. (4) and factoring out the coarse mass gives

$$\begin{aligned} \widetilde{o}_{T,t} &= \bar{o}_{T,t} + w_t d_h^v, & w_t &= \frac{a_t e^{d_h^\ell}}{1 - a_t + a_t e^{d_h^\ell}}, \\ \widetilde{\ell}_{T,t} &= \bar{\ell}_{T,t} + \log\left(1 - a_t + a_t e^{d_h^\ell}\right). \end{aligned} \tag{16}$$

The value weight w_t reflects both the drift weight and the cached mass correction. When $a_t = 1$, the blend uses the full refinement. As a_t approaches zero, both the value and log-mass approach their fresh coarse estimates. With current residuals $d_t^v = o_{T,t} - \bar{o}_{T,t}$ and $d_t^\ell = \ell_{T,t} - \bar{\ell}_{T,t}$, the errors become

$$\begin{aligned} \widetilde{o}_{T,t} - o_{T,t} &= w_t d_h^v - d_t^v, \\ \widetilde{\ell}_{T,t} - \ell_{T,t} &= \log\left(1 - a_t + a_t e^{d_h^\ell}\right) - d_t^\ell. \end{aligned} \tag{17}$$

These expressions show how drift weighting regulates the reused residual while retaining the current coarse component.

To relate the tail errors to the full attention output, let $\lambda_t = e^{\ell_{T,t}}/(e^{\ell_{H,t}} + e^{\ell_{T,t}})$ be the exact fraction of mass assigned to the tail, and let $\widetilde{\lambda}_t$ use $\widetilde{\ell}_{T,t}$ in place of $\ell_{T,t}$. Since the selected branch is computed exactly, the reconstructed output satisfies

$$\begin{aligned}\widetilde{o}_t - o_t = \widetilde{\lambda}_t\,(\widetilde{o}_{T,t} - o_{T,t}) \\ + \left(\widetilde{\lambda}_t - \lambda_t\right)(o_{T,t} - o_{H,t})\,.\end{aligned} \tag{18}$$

The first term carries the tail value error, while the second captures the change in weighting between the selected branch and the tail. This separates the two effects that value and mass refinement address.

B Prediction Reuse and Online Scheduling

Computed predictions provide both sources for prediction reuse and feedback on its accuracy. Dense refreshes additionally evaluate attention refinement. We use these observations to fit the extrapolation gain and coordinate renewal at both levels.

B.1 Online Prediction Update

Online gain estimation. For consecutive computed steps $r < s < t$, the observed prediction change is $y_t = p_t - p_s$, while the corresponding unit-gain extrapolation is $z_t = (t-s)(p_s - p_r)/(s-r)$. The set $\mathcal{C}_t$ collects these observations through step t whenever two preceding sources are available, including observations from both dense and corrected sparse evaluations. Setting the derivative of the least-squares objective in Eq. (6) to zero gives

$$\alpha_t = \frac{\sum_{j\in\mathcal{C}_t}\langle y_j, z_j\rangle_F}{\sum_{j\in\mathcal{C}_t}\|z_j\|_F^2}. \tag{19}$$

The numerator measures agreement between observed and extrapolated changes, while the denominator normalizes by the accumulated squared magnitude of the extrapolated changes. The gain starts at zero and remains zero until the denominator is positive.

At each computed step, we first evaluate the mismatch between p_t and the prediction formed from the previous history and gain. This supplies the scheduling feedback in Sec. 3.4 before the new observation updates the fit. The computed prediction then becomes the new anchor, and the velocity is updated to $(p_t - p_s)/(t-s)$. Reuse holds the latest prediction when only one source is available and otherwise follows Eq. (5), leaving the history and gain unchanged. When guidance uses separate prediction branches, each branch retains its own history, while their inner products and squared norms are pooled to fit a shared gain.

Source-error decomposition. Let $p_j^D = f_\theta(x_j, j, c)$ denote the dense prediction on the latent state reached by the same sampling run at step j. For a computed source p_j, define its error as $\epsilon_j = p_j - p_j^D$. For reuse at step t from sources $r < s$, write $\kappa_{s,t} = \alpha_s(t-s)/(s-r)$ using the gain fitted by that run. Substituting into Eq. (5) gives

$$\begin{aligned}\widehat{p}_t - p_t^D = \underbrace{p_s^D + \kappa_{s,t}(p_s^D - p_r^D) - p_t^D}_{\text{extrapolation from dense sources}} \\ + \underbrace{\epsilon_s + \kappa_{s,t}(\epsilon_s - \epsilon_r)}_{\text{source errors carried into reuse}}\,.\end{aligned} \tag{20}$$

The source errors affect both the latest anchor through ϵ_s and the extrapolated change through $\kappa_{s,t}(\epsilon_s - \epsilon_r)$. Their contribution therefore depends on both their magnitude and how they change between sources, while the total reuse error also depends on their alignment with the extrapolation error from dense sources. Corrected sparse evaluations renew the anchor and velocity between dense refreshes, connecting attention refinement to both components of prediction reuse.

B.2 Error-Guided Scheduling

The scheduler combines error-based reuse allowances with the remaining computation budget.

Prediction feedback. At each computed step, the mismatch in Eq. (7) evaluates the extrapolated estimate before its source history is updated. The proposed computation interval G starts at two sampler steps. For an observed interval $g = t - s \geq 2$, we set $G = g + 2$ if the mismatch rate is below τ_{pred}, and $G = g$ otherwise. One-step observations leave G unchanged. The pacing cap below can shorten the proposed interval according to the remaining budget and error rate.

Refinement feedback. Dense refreshes provide the reference needed to evaluate attention refinement through Eq. (8). Error below τ_{ref} at age a permits the new refinement to be reused up to age $a + L$. The extension L starts at two steps and grows by two after each pass. A failed check revokes the reuse allowance and resets L to two, so the next computed prediction uses dense attention. A newly initialized refinement is likewise checked at the next computed step, and computed steps remain dense until a check passes. This feedback determines where sparse reconstruction is eligible.

Budget-based coordination. Let R_t and $K_t > 0$ be the remaining sampler steps and computations. Their ratio gives the baseline spacing q_t. With ρ_s the latest prediction-error rate and $\bar{\rho}_s$ its mean since prediction reuse began, define

$$\begin{aligned} q_t &= \max\left\{1, \left\lceil \frac{R_t}{K_t} \right\rceil\right\}, \\ G_t^{\text{pace}} &= \min\left\{q_t, \max\left\{1, \text{round}\left[q_t \frac{\bar{\rho}_s}{\rho_s}\right]\right\}\right\}. \end{aligned} \tag{21}$$

The baseline is recomputed as the remaining budget changes. The error-rate ratio shortens the spacing when the latest mismatch exceeds its running mean, while the outer cap prevents spacing beyond q_t. Before positive error-rate statistics are available, the cap equals q_t.

A new prediction is computed when its age reaches the smaller of the feedback-proposed interval and G_t^{pace}. It uses corrected sparse attention while refinement remains eligible and dense attention otherwise. Every computed prediction consumes one unit of B. Algorithm 1 summarizes the resulting three-action schedule and state updates.

Algorithm 1 RECAP sampling with two levels of reuse

Require: Initial latent x_0, conditioning c, T steps, budget B, error tolerances $\tau_{\text{pred}}, \tau_{\text{ref}}$

Initialize empty refinement $\mathcal{D}$ and history $\mathcal{P}$, gain $\alpha = 0$, reuse allowances, and budget
for $t = 0, \ldots, T-1$ **do**
  Select prediction reuse, sparse, or dense evaluation under reuse allowances and budget
  if the action is prediction reuse **then**
    $p_t \leftarrow \widehat{p}_t$ from Eq. (5)
  else
    if the action is a dense refresh **then**
      Compute p_t, pool errors of any old refinement using Eq. (8), and renew it
    else
      Compute p_t using corrected sparse attention
    end if
    Measure mismatch using Eq. (7) if history exists, then update reuse intervals
    Update the gain using Eq. (19) when two prior sources exist
    Renew prediction history and velocity with p_t, and decrease the remaining budget
  end if
  $x_{t+1} \leftarrow \Psi_t(x_t, p_t)$
end for
return x_T

C Implementation Details

C.1 Motivation Experiments

Composition and error reference. The composition experiments match prompts, initial noise, and sampling settings across sparse attention, temporal caching, and their combination. Figure 4a reports the resulting speed–fidelity trade-off for DFSAttn, TeaCache, and their composition, with ten settings per family, each averaged over three prompts. To define the error reference in Fig. 4b, let Y_D, Y_S, Y_C, Y_{SC} denote the concatenated decoded frames from dense, sparse-only, cache-only, and composed generation, respectively, and write

$$e_S = Y_S - Y_D, \qquad e_C = Y_C - Y_D, \qquad e_{SC} = Y_{SC} - Y_D, \qquad \rho = \frac{\|e_{SC}\|_2}{\|e_S + e_C\|_2}. \tag{22}$$

The denominator retains the alignment of the standalone errors, with each output obtained from a complete closed-loop run that includes accumulated trajectory differences. The plotted groups contain 30, 26, 18, and 30 configurations at 10, 15, 20, and 25 reused steps, and the composed error falls below the reference in 103 of these 104 configurations. These counts include repeated settings of the same prompt. The ratio summarizes complete sampling outcomes, while the cache-fill intervention below isolates the effect of changing the information supplied for reuse.

Intervening on cache sources. To isolate the role of source accuracy, the intervention in Fig. 4c changes the tensors written into the cache while fixing the sparse-attention settings and reuse schedule. At each source step, dense and sparse evaluations share the run's current input, with the dense evaluation supplying the tensors stored by TeaCache and the sparse prediction advancing the sampler. We measure the resulting latent deviation from the matched dense trajectory, so the comparison isolates source accuracy at unchanged computation locations.

Persistence of sparsification error. To examine whether sparsification error has reusable structure, the diagnostic in Fig. 4d evaluates dense and sparse predictions on identical latent inputs at each step along a dense CogVideoX-1.5 trajectory. Their difference $\epsilon_t = p_t^S - p_t^D$ isolates sparsification error from accumulated trajectory drift. We measure its directional alignment between a source step s and a subsequent reuse step t using

$$\cos(\epsilon_s, \epsilon_t) = \frac{\langle \epsilon_s, \epsilon_t \rangle}{\|\epsilon_s\|_2 \|\epsilon_t\|_2}. \tag{23}$$

For each TeaCache threshold $\tau \in \{0.10, 0.20, 0.30\}$, we derive a common reuse schedule from one reference prompt. This schedule identifies the source–reuse pairs to compare, with s the latest source step preceding t. We evaluate the same pairs along each prompt's own dense trajectory and average over ten prompts. The consistently high cosine similarity indicates that sparsification errors retain similar directions across nearby steps, motivating the reuse of correction information.

C.2 Attention Computation

Covariance representation. To evaluate the moment estimates in Appendix Sec. A.1, we represent key variation in a shared covariance basis. Let u_j be the eigenvectors of $N^{-1} \sum_{b \in \mathcal{B}} n_b C_b$ in decreasing eigenvalue order, and let $w_{b,j} = u_j^\top C_b u_j$ be the projected variance of block b. We retain the leading $R = 4$ directions and represent the remaining variation by its average,

$$\widetilde{w}_{b,j} = \begin{cases} w_{b,j}, & j \le R, \\ (d_k - R)^{-1} \sum_{r>R} w_{b,r}, & j > R, \end{cases} \qquad \nu_b(q) = \beta^2 \sum_j \widetilde{w}_{b,j} (u_j^\top q)^2. \tag{24}$$

This approximation preserves the covariance trace while retaining its leading directional variation. The projected variances and shared value-key covariance are recomputed from current inputs at each sparse evaluation.

Moment evaluation. The moment expansion is applied where the estimated logit variance is moderate. Using the fixed indicator $g_b(q) = \mathbf{1}[\nu_b(q) \le \theta^2]$ with $\theta = 2$, the evaluated block estimates are

$$L_b = \log n_b + \beta q^\top \bar{k}_b + \tfrac{1}{2} g_b(q) \nu_b(q), \qquad \bar{v}_b^{\text{corr}} = \bar{v}_b + g_b(q) \delta v(q). \tag{25}$$

When the indicator is zero, these reduce to block-mean estimates, and Eq. (13) aggregates them.

C.3 Memory Optimization

Attention refinement is accessed only during dense refreshes and corrected sparse evaluations, one layer at a time. We therefore store it in host memory and transfer it through two GPU staging buffers, prefetching the next layer's refinement while the current layer computes. This changes memory placement without altering the computation. Frame hashes for ten paired prompts across the three models exactly match those from device-resident runs.

Table 4 compares peak GPU memory with sparse–cache compositions. RECAP falls within their memory range on CogVideoX-1.5 and Wan2.1-1.3B. On HunyuanVideo, it uses less memory than every composition and remains within 1 GiB of dense generation. Relative to device-resident execution, streaming adds 0.3% latency on CogVideoX-1.5, no measurable overhead on Wan2.1-1.3B, and at most 1% on HunyuanVideo. These measurements use paired streamed and device-resident runs executed concurrently on two GPUs with the same prompts. All reported latencies include transfer overhead. Peak memory depends on cached states, temporary attention buffers, and model offloading. On Wan2.1-1.3B, several methods remain near the approximately 11 GiB peak associated with text encoding.

Table 4: Peak allocated GPU memory in GiB. All methods run with the same model offloading.

Method	**CogVideoX-1.5**	**Wan2.1-1.3B**	**HunyuanVideo**
Dense	17.20	10.83	36.30
DFSAttn + NaviCache	20.70	10.83	40.52
DFSAttn + LeMiCa	21.62	10.83	42.49
DFSAttn + TeaCache	21.62	10.83	43.17
SVOO + NaviCache	22.02	14.08	47.50
SVOO + LeMiCa	22.94	14.08	47.50
SVOO + TeaCache	22.94	14.08	47.50
RECAP	20.80	10.99	37.25

C.4 Parameter Selection

Each operating point is specified by a computation budget B and two error tolerances, τ_{pred} and τ_{ref}. For all models, we select the tolerances from $\{0.025, 0.05, 0.10, 0.20\}$, while budgets are chosen separately for each model to account for differences in inference cost.

We first choose B to approximately match the target latency. Using two fixed prompts per model and their dense-generation references, we then compare tolerance settings by varying one threshold at a time, with PSNR, SSIM, and LPIPS measuring fidelity to the dense outputs. We retain settings that preserve fidelity within the target latency range and adjust B if necessary, since the tolerances also affect runtime. This sequential procedure avoids an exhaustive search over all parameter combinations. The selected triplets in Tab. 5 are then fixed for all evaluation prompts, without per-prompt tuning.

Table 5: Operating-point settings selected using the procedure in Appendix Sec. C.4. Each triplet is fixed across evaluation prompts.

Configuration	B	τ_{pred}	τ_{ref}
CogVideoX Quality	32	0.025	0.1
CogVideoX Speed	30	0.025	0.1
CogVideoX Fast	18	0.05	0.05
Wan2.1 Quality	26	0.025	0.1
Wan2.1 Speed	20	0.05	0.1
Wan2.1 Fast	15	0.1	0.05
HunyuanVideo Quality	31	0.1	0.2
HunyuanVideo Speed	24	0.1	0.2
HunyuanVideo Fast	17	0.2	0.2

D IMAGE GENERATION

We further evaluate RECAP for text-to-image generation on FLUX.1-dev (Black Forest Labs, 2024) at 1024×1024 resolution, using 50 denoising steps and a guidance scale of 3.5. We compare RECAP with the caching methods LeMiCa (Gao et al., 2025) and Spectrum (Han et al., 2026) at similar latency. Using the DrawBench prompts (Saharia et al., 2022), we measure fidelity to dense generation with PSNR (Horé & Ziou, 2010), SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). We assess prompt-conditioned preference with ImageReward (Xu et al., 2023). As shown in Tab. 6, RECAP is the fastest method and achieves the best fidelity among the accelerated methods. It also retains an ImageReward score close to dense generation with less than 0.1 GiB of additional peak memory. Figure 7 shows that RECAP also follows the dense composition more closely.



Figure 7: Results on FLUX.1-dev. RECAP follows the composition of dense generation more faithfully than the baselines, for example in the shape of the umbrellas and the red dress of the couple in the third column and in the buildings and pavement behind Mario in the first column.

Table 6: Image-generation results. Experiments are conducted on one NVIDIA A800 GPU.

Method	PSNR↑	SSIM↑	LPIPS↓	ImageReward↑	Latency/s↓	Speedup↑	Mem./GiB↓
Dense	–	–	–	1.0762	25.84	1.00×	33.83
LeMiCa (NeurIPS'25)	24.87	0.8706	0.1414	1.0471	7.59	3.40×	**33.85**
Spectrum (CVPR'26)	<u>25.68</u>	<u>0.8778</u>	<u>0.1166</u>	<u>1.0487</u>	<u>7.53</u>	<u>3.43×</u>	34.27
RECAP	**26.90**	**0.9006**	**0.1019**	**1.0712**	**7.21**	**3.58×**	<u>33.91</u>

E ADDITIONAL RESULTS

E.1 RESULTS ON WAN2.1-1.3B

Tab. 7 reports the normal-regime comparison with individual accelerators and the high-speed comparison with direct sparse–cache compositions on Wan2.1-1.3B.

Table 7: Wan2.1-1.3B results. Columns and evaluation metrics follow the main comparisons. RECAP-SPEED and RECAP-QUALITY denote the speed- and quality-oriented settings.

Method	PSNR↑	SSIM↑	LPIPS↓	CLIPIQA+↑	VBench↑	Latency↓	Speedup↑
Wan2.1-1.3B (480×832, 81 frames)							
Dense	–	–	–	0.531	83.39	273s	1.00×
TeaCache (CVPR'25)	24.56	0.8789	0.0876	0.535	83.22	162s	1.69×
SVG (ICML'25)	17.74	0.7018	0.2773	0.422	81.40	175s	1.56×
Re-ttention (NeurIPS'25)	26.16	0.9095	0.0664	0.485	81.83	225s	1.21×
LeMiCa (NeurIPS'25)	29.51	0.9456	0.0426	0.535	83.22	163s	1.67×
SVOO (ICML'26)	23.75	0.8676	0.1015	0.500	82.95	231s	1.18×
DFSAttn (ICML'26)	20.08	0.7668	0.2158	0.396	81.54	186s	1.47×
NaviCache (ICML'26)	25.13	0.8753	0.0864	0.532	83.22	138s	1.98×
RECAP-SPEED	27.05	0.9141	0.0741	0.540	83.39	**131s**	**2.08**×
RECAP-QUALITY	**30.29**	**0.9520**	**0.0372**	**0.545**	**83.49**	158s	1.73×
DFSAttn + TeaCache	18.62	0.7143	0.2409	0.401	80.80	112s	2.44×
DFSAttn + LeMiCa	18.65	0.7303	0.2196	0.419	81.31	118s	2.31×
DFSAttn + NaviCache	18.52	0.7146	0.2299	0.402	80.90	118s	2.31×
SVOO + TeaCache	20.73	0.7894	0.1561	0.493	82.06	120s	2.27×
SVOO + LeMiCa	22.04	0.8281	**0.1210**	0.500	82.22	115s	2.37×
SVOO + NaviCache	21.82	0.8182	0.1253	0.491	82.09	111s	2.46×
RECAP	**23.51**	**0.8323**	0.1238	**0.534**	**82.88**	**105s**	**2.60**×

E.2 DETAILED VBENCH RESULTS

Tab. 8 and Tab. 9 report the evaluated VBench quality dimensions for the normal-regime and direct-composition comparisons, respectively. IQ denotes imaging quality, SC subject consistency, BC background consistency, AQ aesthetic quality, MS motion smoothness, and TF temporal flickering. Overall is the mean of the six scores after per-dimension normalization, $\frac{100}{6}\sum_d (s_d - m_d)/(M_d - m_d)$, where s_d is the raw score on a $[0, 1]$ scale. In the order IQ, SC, BC, AQ, MS, and TF, the normalization bounds (m_d, M_d) are $(0, 1)$, $(0.1462, 1)$, $(0.2615, 1)$, $(0, 1)$, $(0.7060, 0.9975)$, and $(0.6293, 1)$ from the official VBench implementation.

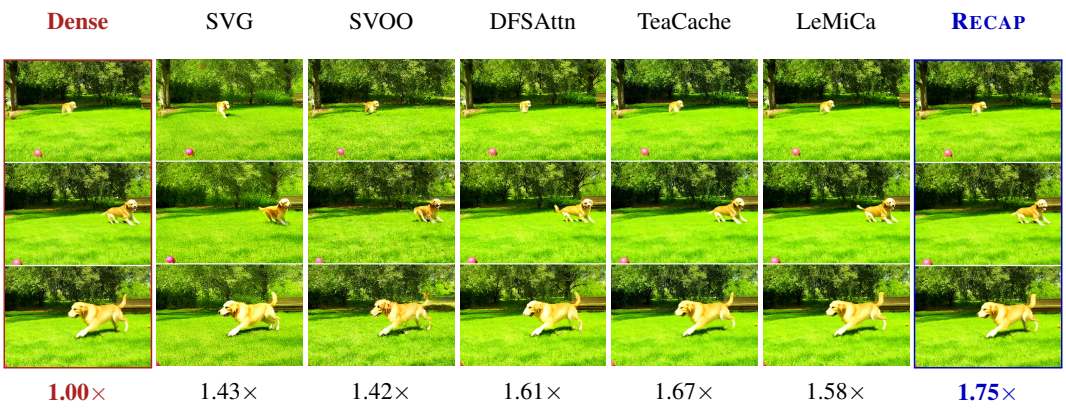
Table 8: VBench quality dimensions for the normal-regime comparison with individual methods in Tab. 1 and the upper part of Tab. 7. RECAP-SPEED and RECAP-QUALITY denote the speed- and quality-oriented settings. Best and second-best results are bold and underlined.

Method	IQ↑	SC↑	BC↑	AQ↑	MS↑	TF↑	Overall↑
CogVideoX-1.5 (768×1360, 81 frames)							
Dense	64.05	93.70	96.19	61.33	98.27	97.06	83.31
TeaCache (CVPR'25)	63.16	93.65	96.17	60.95	98.24	97.06	83.06
SVG (ICML'25)	63.41	93.34	96.07	60.16	98.13	96.76	82.69
Re-ttention (NeurIPS'25)	57.67	93.24	96.12	56.38	98.06	96.98	81.16
LeMiCa (NeurIPS'25)	63.31	93.57	96.17	61.05	98.24	97.07	83.09
SVOO (ICML'26)	63.50	93.33	96.04	60.12	98.11	96.91	82.75
DFSAttn (ICML'26)	63.32	93.56	96.14	60.58	98.11	96.88	82.85
NaviCache (ICML'26)	63.22	93.65	96.16	60.68	98.23	**97.10**	83.04
RECAP-SPEED	63.56	93.59	**96.18**	61.08	**98.25**	97.09	83.16
RECAP-QUALITY	**63.79**	**93.66**	96.16	**61.28**	**98.25**	97.06	**83.23**
Wan2.1-1.3B (480×832, 81 frames)							
Dense	66.01	93.75	95.60	63.30	97.95	96.48	83.39
TeaCache (CVPR'25)	64.83	93.69	**95.57**	63.04	98.02	96.58	83.22
SVG (ICML'25)	63.22	92.66	94.84	60.24	97.50	95.65	81.40
Re-ttention (NeurIPS'25)	63.56	91.34	94.56	63.06	97.59	96.04	81.83
LeMiCa (NeurIPS'25)	64.98	93.66	95.45	63.11	98.02	96.57	83.22
SVOO (ICML'26)	65.48	93.42	95.44	62.83	97.86	96.20	82.95
DFSAttn (ICML'26)	63.40	92.57	95.13	60.05	97.54	95.79	81.54
NaviCache (ICML'26)	65.54	93.64	95.46	62.59	98.00	96.59	83.22
RECAP-SPEED	65.62	93.71	95.47	63.14	**98.10**	96.57	83.39
RECAP-QUALITY	**65.79**	**93.81**	95.51	**63.21**	98.04	**96.72**	**83.49**
HunyuanVideo (720×1280, 129 frames)							
Dense	64.91	96.33	96.88	58.74	99.33	98.92	85.13
TeaCache (CVPR'25)	62.82	96.11	96.77	57.96	99.30	98.92	84.57
SVG (ICML'25)	63.76	95.55	96.48	58.26	99.28	98.76	84.51
LeMiCa (NeurIPS'25)	63.12	96.34	96.90	58.45	99.38	99.03	84.87
SVOO (ICML'26)	64.11	96.19	96.76	58.40	99.28	98.91	84.85
DFSAttn (ICML'26)	64.14	96.20	96.68	58.45	99.25	98.80	84.78
NaviCache (ICML'26)	**64.63**	96.33	96.63	58.68	99.32	98.94	85.02
RECAP-SPEED	63.82	96.32	96.90	58.66	**99.40**	**99.07**	85.04
RECAP-QUALITY	64.27	**96.36**	**96.97**	**58.73**	99.36	98.96	**85.08**

Table 9: VBench quality dimensions for the high-speed comparison with direct compositions in Tab. 2 and the lower part of Tab. 7. Best and second-best results are bold and underlined.

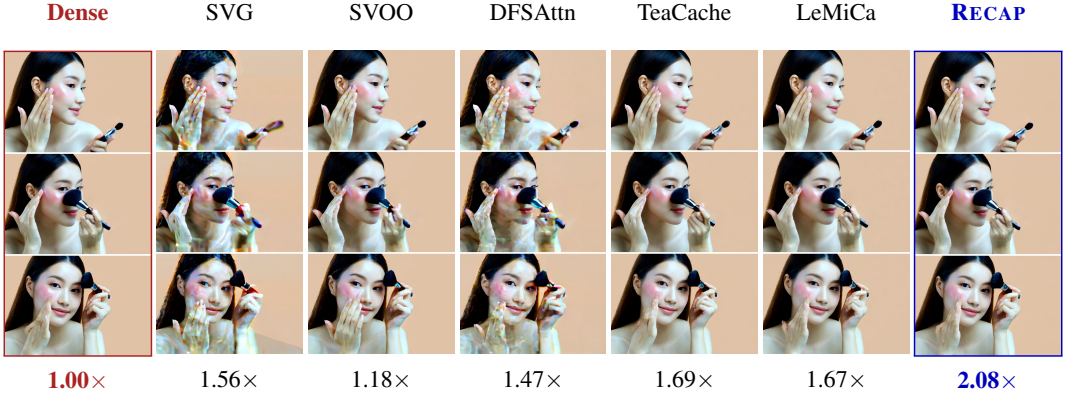
Method	IQ↑	SC↑	BC↑	AQ↑	MS↑	TF↑	Overall↑
CogVideoX-1.5 (768×1360, 81 frames)							
DFSAttn + TeaCache	61.59	93.71	96.27	58.32	97.64	95.93	81.55
DFSAttn + LeMiCa	63.43	93.90	96.30	60.76	97.65	95.91	82.30
DFSAttn + NaviCache	63.39	93.64	96.20	60.42	97.67	96.04	82.23
SVOO + TeaCache	62.79	93.69	96.12	56.69	97.67	95.83	81.41
SVOO + LeMiCa	63.34	93.60	96.28	60.34	97.81	96.07	82.32
SVOO + NaviCache	62.95	93.23	95.98	58.34	97.63	96.17	81.72
RECAP	**63.61**	**93.95**	**96.36**	**60.86**	**97.82**	**96.30**	**82.64**
Wan2.1-1.3B (480×832, 81 frames)							
DFSAttn + TeaCache	63.56	90.42	94.35	61.13	97.22	95.42	80.80
DFSAttn + LeMiCa	64.26	90.86	94.78	61.98	97.30	95.48	81.31
DFSAttn + NaviCache	64.20	90.68	94.51	60.97	97.16	95.35	80.90
SVOO + TeaCache	64.16	91.67	94.63	63.29	97.62	96.02	82.06
SVOO + LeMiCa	64.55	91.46	94.94	63.46	97.65	96.06	82.22
SVOO + NaviCache	64.71	91.23	**95.15**	62.98	97.58	95.97	82.09
RECAP	**65.71**	**92.47**	94.95	**63.60**	**97.80**	**96.42**	**82.88**
HunyuanVideo (720×1280, 129 frames)							
DFSAttn + TeaCache	**62.71**	94.93	96.20	58.14	99.15	98.43	83.91
DFSAttn + LeMiCa	60.69	**95.34**	96.19	57.80	99.28	98.69	83.79
DFSAttn + NaviCache	62.18	95.00	95.93	58.14	99.16	98.48	83.80
SVOO + TeaCache	60.95	94.65	96.27	57.85	99.11	98.45	83.52
SVOO + LeMiCa	61.18	95.17	96.12	57.97	99.26	98.73	83.86
SVOO + NaviCache	62.31	95.04	95.95	58.35	99.20	98.64	83.97
RECAP	62.35	95.21	**96.49**	**58.68**	**99.33**	**98.81**	**84.34**

E.3 More Visual Results



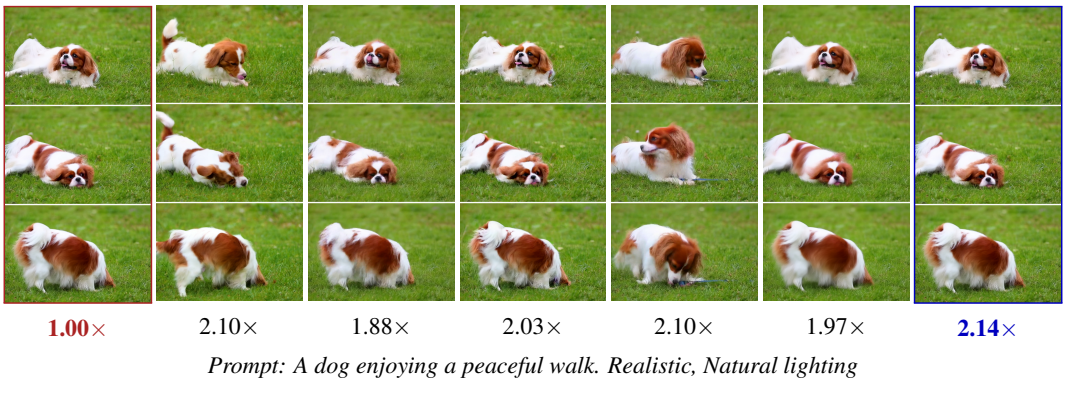
Prompt: A playful golden retriever bounds through a sunlit park, its fur gleaming in the afternoon light. The...

Figure 8: Results on CogVideoX-1.5. RECAP better preserves the dog's posture, including its tail and legs. Sparse-attention baselines shift the dog's motion, and SVG and SVOO distort its face.



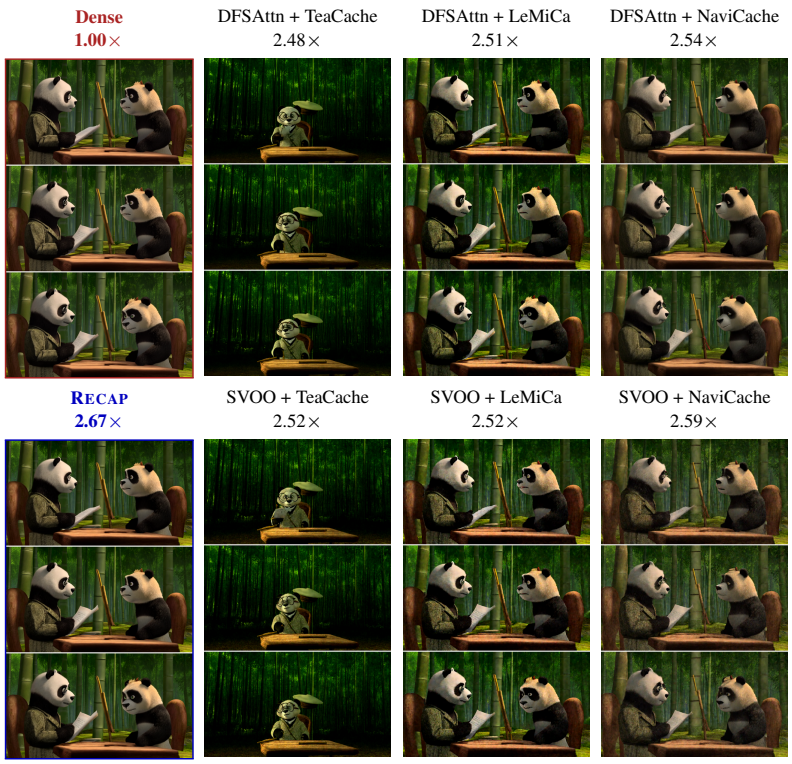
Prompt: A young woman with flawless skin and a serene expression sits at a vanity, bathed in soft morning...

Figure 9: Results on Wan2.1-1.3B. RECAP preserves the hands, makeup brush, and facial details. SVG and DFSAttn add colored artifacts to the hands and skin, and SVOO alters the hand pose.



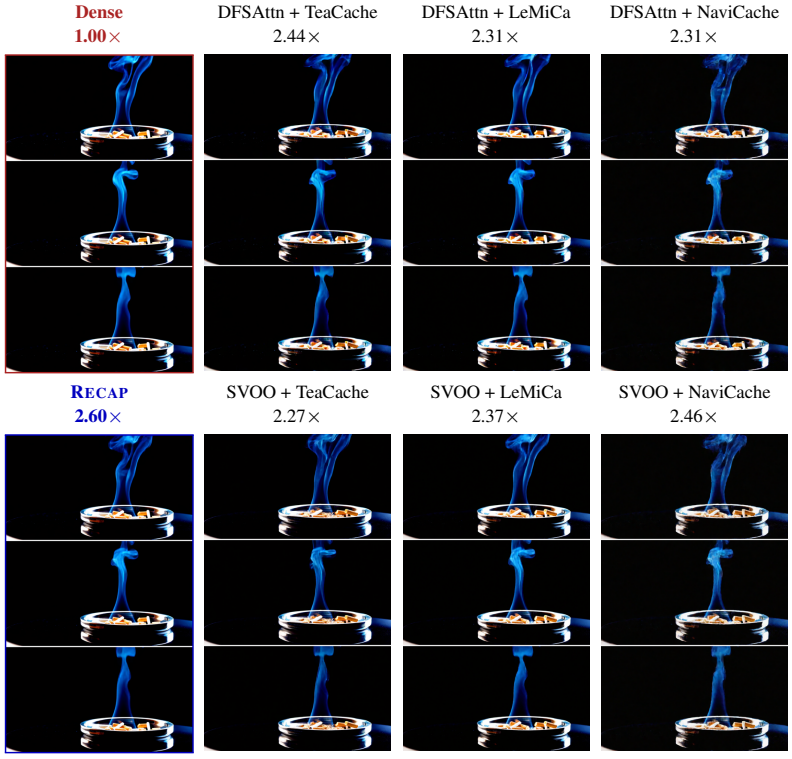
Prompt: A dog enjoying a peaceful walk. Realistic, Natural lighting

Figure 10: Results on HunyuanVideo. RECAP preserves the pose of the dog and its coat pattern across frames. SVG and TeaCache change the dog's motion, while LeMiCa and SVOO blur its fur.



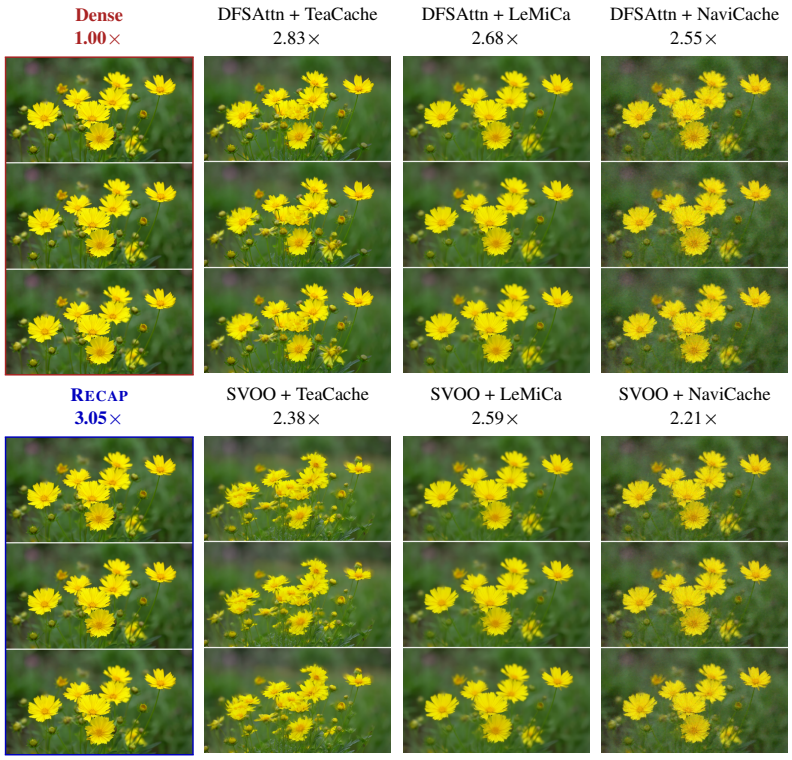
Prompt: In a serene bamboo forest, two pandas sit at a rustic wooden table, surrounded by lush greenery....

Figure 11: Results on CogVideoX-1.5 with compositions. RECAP preserves both pandas, while the TeaCache compositions reduce the scene to one figure and most others alter the right panda's face.



Prompt: A close-up shot reveals an ashtray brimming with cigarette butts, each one a testament to moments...

Figure 12: Results on Wan2.1-1.3B with compositions. RECAP reproduces the rising smoke and cigarette butts, while the NaviCache compositions add grain and some others alter the smoke's curl.



Prompt: Yellow flowers sway in the wind. Realistic, Natural lighting, Peaceful

Figure 13: Results on HunyuanVideo with compositions. RECAP keeps the flower arrangement and petal details, while the TeaCache compositions rearrange flowers and the NaviCache ones add grain.

E.4 MORE ABLATIONS

Tab. 10 repeats the ablations of Sec. 4.3 at the CogVideoX Fast operating point in Tab. 5, which is the setting of RECAP in the comparison with direct compositions in Tab. 2. All variants use the same controls and prompt panels as the normal-regime ablations in Tab. 3.

Table 10: Ablations on CogVideoX-1.5 at the fast operating point that RECAP uses in Tab. 2. Latency is measured on one NVIDIA A800 GPU. Bold marks the best value within each comparison.

Variant	PSNR↑	SSIM↑	LPIPS↓
W/o refinement	24.91	0.8592	0.1478
Frozen state	10.42	0.3318	0.7935
RECAP	**25.68**	**0.8768**	**0.1296**

(a) Reconstruction components.

	Same locations			*Matched runtime*		
Variant	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PISA	22.61	0.7725	0.2481	22.62	0.7735	0.2432
Re-ttention	23.54	0.7795	0.2270	24.70	0.8194	0.1850
RECAP	**25.68**	**0.8768**	**0.1296**	**25.68**	**0.8768**	**0.1296**

(b) Alternative reconstructions.

Variant	PSNR↑	SSIM↑	LPIPS↓
Dense-only history renewal	24.20	0.8544	0.1576
Uniform schedule	24.61	0.8517	0.1585
RECAP	**25.68**	**0.8768**	**0.1296**

(c) Prediction-history renewal and online scheduling.

Variant	PSNR↑	SSIM↑	LPIPS↓	Latency
W/o prediction reuse	**25.71**	0.8709	0.1336	748.86s
RECAP	25.68	**0.8768**	**0.1296**	**391.16s**

(d) Prediction reuse.

The conclusions match the normal regime. Removing the refinement lowers PSNR from 25.68 to 24.91 dB. Freezing the initial refinement and partition collapses it to 10.42 dB. At matched runtime, RECAP leads adapted PISA and Re-ttention by 3.06 and 0.98 dB. Restricting history renewal to dense

refreshes lowers PSNR to 24.20 dB. Prediction reuse nearly halves latency, from 748.86 to 391.16 s, and changes PSNR by only 0.03 dB. Uniform placement lowers PSNR by 1.07 dB. This drop is larger than in the normal regime because placement matters more when computation is scarce.

F Extended Related Work

Approximating omitted attention. PISA (Li et al., 2026) and COBS (Tian et al., 2026) estimate the contribution of omitted key blocks from block statistics, using first-order value compensation and second-order mass estimation, respectively. These moment estimates are recomputed at every step, but they cannot recover within-block structure beyond the modeled moments. RECAP uses them only as its fresh coarse component and adds a cached dense residual for the remaining discrepancy, while drift weighting limits this residual as it ages. Replacing our reconstruction with adapted PISA within the same reuse framework lowers PSNR by 4.31 dB at matched runtime, as Tab. 3b shows.

Attention reuse across steps. Re-ttention (Chen et al., 2025) reconstructs sparse attention from normalization ratios and output residuals cached at earlier steps, and DiTFastAttn (Yuan et al., 2024) shares attention residuals and outputs across steps under an offline compression plan. RECAP also reuses attention information across steps, but it combines the cached residual with fresh coarse statistics, regulates it by the measured drift, and verifies it against dense attention at each refresh. At matched runtime, replacing our reconstruction with adapted Re-ttention lowers PSNR by 7.47 dB. Moreover, these methods reduce computation only within attention modules, whereas a prediction-reuse step in RECAP skips the entire denoiser, including projections and feed-forward layers.

Coupling sparse attention and caching. LoSA (Liu et al., 2026b) reuses an early sparse mask and studies composition with feature caching while keeping the cache schedule, predictor, and residuals unchanged. Like the direct compositions in Tab. 2, it treats the two approximations as independent components. In RECAP, corrected sparse predictions renew the sources for prediction reuse, so one dense refresh supports several sparse evaluations and their subsequent reuse. This coupling allows RECAP to achieve higher PSNR than all six direct compositions on all three video models while running faster.

Scheduling. Several methods determine their schedules through offline search. DiTFastAttn greedily tries candidate compression strategies for every layer at every step, running a forward pass of the model for each candidate (Yuan et al., 2024). ASTRAEA searches token budgets with an evolutionary algorithm that scores every candidate by generating full videos (Liu et al., 2026c). LeMiCa likewise optimizes its caching schedule offline (Gao et al., 2025), while online caching methods such as TeaCache (Liu et al., 2025a) adapt reuse from changes in model inputs. RECAP also adapts online, but it measures errors on the quantities it reuses and requires no search over execution locations. Only the budget and two error tolerances are selected in advance, using a small number of runs on two prompts per model as described in Appendix Sec. C.4.

G Limitations

RECAP adapts attention refinement and prediction reuse within a prescribed computation budget. Extending this adaptation to the budget itself could allow computation to vary with prompt complexity and the evolving denoising trajectory. Another direction is finer-grained coordination across layers or attention heads, allowing their different reuse characteristics to guide computation allocation. Our evaluation focuses on image and video diffusion transformers. Extending coordinated attention and prediction caching to streaming generation and other generative architectures would further broaden the framework's scope.