RDA: RECOVERING DROPPED CONTRIBUTIONS FROM RETAINED ATTENTION FOR VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The quadratic cost of self-attention is a major bottleneck in video diffusion transformers. Training-free sparse attention with online top-k selection has emerged as a flexible solution, but existing methods struggle to reconcile mask adaptivity with execution efficiency. Block-wise methods enable efficient regular computation, yet their coarse partitions limit the flexibility of the resulting masks; centroid-wise methods construct more content-adaptive masks, but incur costly preprocessing and irregular computation. We propose RDA, a training-free sparse attention method that retains the efficiency of regular block-wise execution while recovering information lost by coarse block masks. RDA co-designs recovery-aware top-p block selection and compensation to predict dropped contributions from already-computed retained attention outputs. We further introduce a sensitivity-driven two-dimensional scheduler that jointly assigns top-p configurations across denoising steps and network layers, minimizing a calibrated attention-error objective under a target sparsity budget. On 720p HunyuanVideo and Wan2.1, RDA achieves 2.010× and 1.663× speedups at 78.3% and 77.1% sparsity, respectively, while providing the best fidelity among sparse baselines.

1 INTRODUCTION

Diffusion Transformers (DiTs) (Peebles & Xie, 2023) are widely used for video generation, including in Latte (Ma et al., 2025), CogVideoX (Yang et al., 2025b), Movie Gen (Polyak et al., 2025), HunyuanVideo (Kong et al., 2025), and Wan2.1 (Wan Team et al., 2025). These models generate high-resolution, long-duration videos by iteratively denoising tens of thousands of video tokens. The token count grows with both resolution and frame count: a 720p, 129-frame sequence approaches 120K tokens. Since self-attention (Vaswani et al., 2023) scales quadratically with sequence length, it becomes a major inference bottleneck. Although FlashAttention kernels (Dao et al., 2022; Dao, 2023; Shah et al., 2024) improve execution efficiency, dense FlashAttention-3 still accounts for approximately 68.7% of end-to-end generation time on HunyuanVideo. Reducing attention overhead is therefore key to accelerating DiT video generation.

Training-free sparse attention addresses this bottleneck by reducing the number of query–key interactions. Online block selection offers a way to control the trade-off between generation fidelity and computational cost: it organizes keys into candidate blocks, estimates their contributions to each query group, and evaluates attention on a selected subset. The partition affects both selection and execution. Regular blocks map naturally to GPU tiles, whereas content-aligned blocks can make block-level scores more representative of their constituent tokens. These online methods can therefore be grouped by how they partition tokens: *block-wise* methods use position-aligned uniform blocks, while *centroid-wise* methods use content-based clusters.

SpargeAttn (Zhang et al., 2025a) represents the first category. It scores uniform blocks using mean-pooled queries and keys and retains the top-k blocks. The regular structure supports efficient execution, but position-based partitioning does not account for attention content: mean pooling can obscure strong token responses within otherwise low-scoring blocks. SpargeAttn detects heterogeneous blocks through self-cosine similarity and conservatively retains them. This safeguard preserves more computation, but contributions from dropped blocks remain uncompensated.

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.

SVG2 (Yang et al., 2025a) and SVG-EAR (Zhou et al., 2026) represent the second category, clustering query and key tokens into content-aligned variable-length blocks. SVG2 selects blocks using top-p centroid attention scores; SVG-EAR adds error-aware routing and approximates dropped contributions using cluster centroids. These methods operate near 80% sparsity on HunyuanVideo in our evaluation, but incur preprocessing costs. For SVG-EAR, clustering, selection, tensor permutation, index construction, and compensation together account for about 21.6% of generation time. Variable-length execution with FlashInfer (Ye et al., 2025) also complicates static memory planning, efficient use of the Tensor Memory Accelerator (TMA), and load balancing. The measured Model FLOPs Utilization (MFU) remains below 40%, compared with roughly 60% for dense kernels.

Together, these methods expose a fundamental trade-off between mask adaptivity and execution efficiency. Uniform blocks map efficiently to GPU kernels, but their coarse, position-aligned partitioning limits mask adaptivity and can discard important token-level contributions. Content-based clusters produce more adaptive masks, yet clustering, data reordering, and variable-length execution introduce substantial overhead. The key challenge is therefore to preserve the efficiency of regular block-wise execution while compensating for the information lost by its coarse masks. We observe that the retained and dropped attention contributions depend on the same queries, allowing the already-computed retained outputs to serve as predictive features for the omitted contribution. Our key insight is that **a linear readout can approximately recover dropped contributions from retained attention outputs**.

We propose RDA (Recovering Dropped Contributions from Retained Attention), a training-free method that addresses this challenge through recovery-aware sparse attention. Within each attention call, RDA fits a shared per-head linear readout on sampled full-attention rows to predict the dropped-side unnormalized value sums and normalization masses from the retained outputs, then combines both sides to reconstruct the attention result. Because block selection determines what information remains available for prediction, RDA co-designs selection with recovery: KEXACT preserves strong individual-key responses obscured by key pooling, while VAWARE prioritizes value-heterogeneous blocks that are harder to recover. The resulting uniform blocks retain regular GPU execution, with selection and recovery accounting for only 2.90% of generation time on HunyuanVideo. Finally, since a single sparsity configuration cannot capture the varying importance of denoising steps and network layers, RDA uses a sensitivity-driven two-dimensional top-p schedule to allocate a fixed average-density budget toward regions where additional computation most effectively reduces local attention error.

Our main contributions are summarized as follows:

1. We formulate dropped-output recovery as shared linear prediction from retained augmented-output channels and validate the per-head readout on held-out queries.
2. We co-design KEXACT+VAWARE selection and regular block-sparse GPU kernels for efficient retained-side recovery.
3. We introduce a sensitivity-driven two-dimensional schedule that jointly allocates retained density across denoising steps and network layers.
4. On 720p HunyuanVideo and Wan2.1, RDA delivers 2.010× and 1.663× speedups at 78.3% and 77.1% sparsity, respectively, with the best fidelity among the evaluated sparse methods.

2 Related Work

Training-free sparse attention for video DiTs differs in how masks are constructed and whether dropped contributions are compensated. Masks may be predefined, calibrated offline, or selected online. Online block-selection methods further differ in their use of position-aligned uniform blocks or content-based variable-length clusters.

General and dynamic sparse attention. Early methods use fixed patterns (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2021), hashing (Kitaev et al., 2020), or learned content routing (Roy et al., 2020), typically incorporating sparsity during training. For long-context language models, MInference (Jiang et al., 2024) assigns head-specific patterns offline and constructs indices at inference, whereas SeerAttention, Native Sparse Attention, MoBA, and SpargeAttention2 (Gao et al.,

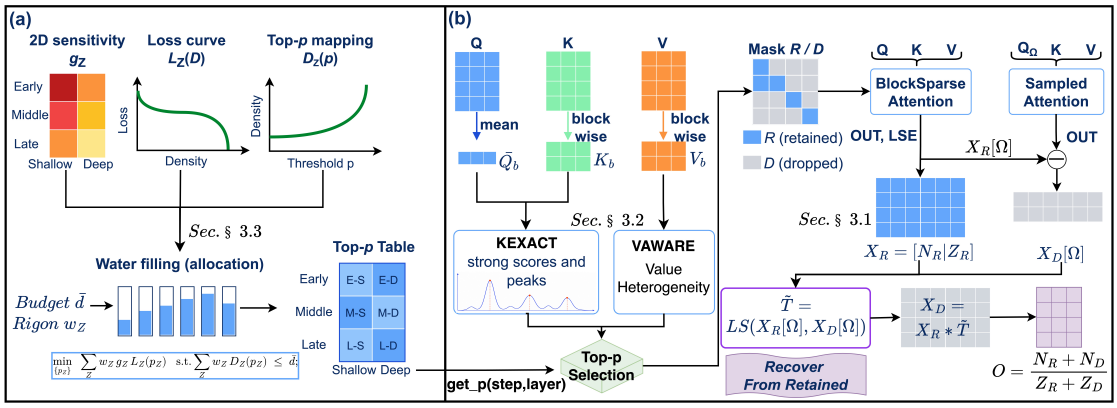

Figure 1: Overview of RDA. (a) Offline, RDA calibrates regional sensitivity and density–loss curves, then allocates a target average-density budget across step–layer regions by water filling to obtain the two-dimensional top-p schedule. (b) Online, the corresponding threshold drives KEXACT+VAWARE block selection; the retained path computes the block-sparse augmented output, and a shared linear readout fitted on sampled full-attention rows predicts the dropped contribution. The two sides are then merged.

2025; Yuan et al., 2025; Lu et al., 2025; Zhang et al., 2026) learn gates or routing policies through training or fine-tuning.

Predefined and calibrated patterns. Sliding Tile Attention (STA) (Zhang et al., 2025b) uses predefined per-head sliding windows that exploit video-token locality. SVG (Xi et al., 2025) selects between cross-frame and intra-frame patterns per head using the MSE of sampled rows. Fixed candidates support optimized kernels but restrict selection to patterns specified before inference. CalibAtt (Yehezkel et al., 2026) instead uses offline calibration to identify reusable sparsity and repetition patterns for video attention.

Dynamic block selection. SpargeAttn (Zhang et al., 2025a) selects uniform blocks by top-k scores between mean-pooled queries and keys, with conservative retention of heterogeneous blocks. SVG2 (Yang et al., 2025a) instead forms content-based variable-length clusters and selects them using top-p centroid attention scores with a cluster-size prior. The two approaches trade off regular execution against content-aligned partitioning.

Compensation for dropped contributions. SVG-EAR (Zhou et al., 2026) extends SVG2 with centroid compensation and selects blocks by approximation error per block area. PISA (Li et al., 2026) uses Taylor expansion to approximate dropped contributions. Both compute approximations from dropped-side key–value information.

RDA predicts dropped contributions from retained augmented outputs using a per-head shared linear readout. It combines this predictor with online top-p selection over uniform blocks and density scheduling across denoising steps and layers.

3 Method

RDA combines shared linear recovery, recovery-oriented block selection, and two-dimensional density scheduling. Section 3.1 introduces the readout model, Section 3.2 describes selection and compensation, and Section 3.3 develops the schedule. Figure 1 summarizes offline calibration and online execution. All derivations are per head; each head is processed independently and uses its own fitted readout.

3.1 Shared Linear Recovery from Retained Outputs

Fix an attention head and a group of B consecutive query rows, with queries $Q \in \mathbb{R}^{B\times d}$, and partition the key–value sequence into a *retained* set R and a *dropped* set D. Fold the softmax scale into Q, and append a constant channel to the values, $\bar{V}_S = [V_S \mid \mathbf{1}]$ for $S \in \{R, D\}$. Define the *augmented output*

$$X_S \;=\; \exp\!\big(QK_S^\top\big)\,\bar{V}_S \;=\; [\,N_S \mid Z_S\,], \qquad S \in \{R, D\}, \tag{1}$$

where N_S contains the unnormalized attention-weighted value sums and Z_S contains the corresponding row-wise normalization masses over S.

Proposition 1 (Additive decomposition)**.** *For each query i in the group, the exact attention output is*

$$o_i \;=\; \frac{N_R[i] + N_D[i]}{Z_R[i] + Z_D[i]}. \tag{2}$$

The block-sparse kernel already computes N_R and Z_R. By Proposition 1, exact attention recovery therefore reduces to recovering $X_D = [N_D \mid Z_D]$.

The Taylor expansion of the exponential defines an exact, infinite-dimensional feature map ϕ: its features consist of tensor powers of the input, with the order-m features scaled by $1/\sqrt{m!}$ for $m = 0, 1, \ldots$. Their inner product therefore satisfies $\langle \phi(q), \phi(k)\rangle = \exp(q^\top k)$. Stacking these features over the rows of Q and K_S yields feature matrices Φ_Q and Φ_{K_S}, in terms of which both sides of equation 1 factor as

$$X_S \;=\; \Phi_Q\, M_S, \qquad M_S \;=\; \Phi_{K_S}^\top \bar{V}_S, \qquad S \in \{R, D\}. \tag{3}$$

This factorization is related to kernelized linear attention (Katharopoulos et al., 2020; Choromanski et al., 2022). For a fixed retained–dropped partition, both outputs are driven by the same query features, combined through different key–value aggregates. This shared structure motivates using the retained responses as predictive features for the dropped responses.

Motivated by the shared query dependence in equation 3, RDA approximates each dropped-side channel as a linear combination of retained-side channels, yielding the matrix readout $\hat{X}_D^{(g)} = X_R^{(g)} T$, where g indexes query groups. The transform T acts along the output-channel dimension and is applied independently to each query row. It thereby uses the retained output to predict the augmented quantities required for reconstruction in Proposition 1.

Rather than fitting a separate transform for each group, RDA fits one transform per head within each attention call from sampled rows pooled across groups. Pooling incorporates their different retained-block selections into a common predictor while avoiding a separate fit for every group. Appendix A defines the recovery metric and compares an interleaved-row holdout (H0) with a whole-group holdout (H1), which removes all rows of selected 128-query groups from fitting. Median capture is 73.53% under H0 and 73.82% under H1. The preserved accuracy on entirely unseen groups supports sharing one transform across query groups.

3.2 Co-designed Selection and Compensation

Selection determines both the retained features X_R and the dropped target X_D, coupling block selection with recovery. RDA uses low-cost scores to prioritize blocks for exact computation, then fits a shared readout on sampled rows to compensate for the remaining contributions, as shown in Figure 1(b).

Selection. RDA partitions queries into groups of B rows and keys into blocks of B tokens. Blocks are selected independently for each query group within each head. The mean query $\bar{q}$ serves as a proxy for the group's shared selection decision, while keys and values provide two complementary block scores.

KEXACT: retain strong responses and within-block peaks. For a key block b with keys $\{k_j\}_{j\in b}$, define

$$\mathrm{Score}_b^K \;=\; \log \sum_{j\in b} \exp\!\big(\bar{q}^\top k_j\big). \tag{4}$$

This is the log unnormalized attention mass that the mean query assigns to block b. Unlike mean-key scoring, KEXACT accumulates individual-key responses before taking the logarithm, preserving peaks under the mean-query proxy.

VAWARE: prioritize value-heterogeneous blocks. Let v_j be the value paired with k_j, let $\bar{v}_b$ be the mean of the values in block b, and define the within-block value variance

$$\mathrm{VarV}_b \;=\; \frac{1}{B}\sum_{j\in b}\big\|v_j-\bar{v}_b\big\|^2. \tag{5}$$

The final block score is

$$\mathrm{Score}_b \;=\; \mathrm{Score}_b^{K} \;+\; w\log\big(\mathrm{VarV}_b+\epsilon\big), \tag{6}$$

where $w \geq 0$ weights value variance and $\epsilon > 0$ ensures numerical stability. VAWARE accounts for value heterogeneity that key-response mass alone does not capture. It prioritizes heterogeneous blocks for exact computation, reducing reliance on recovery for their contributions. Appendix B compares the scoring rules at a matched retained budget.

For each query group, RDA softmax-normalizes the combined scores and sorts blocks in descending order. It retains the smallest prefix whose cumulative probability reaches p. The top-p threshold is applied to the normalized block scores.

Compensation. Let Ω contain query rows sampled at stride s. We compute their full augmented output $X[\Omega]$ using all keys and values. Subtracting the retained-side output $X_R[\Omega]$ gives the dropped-side fitting target,

$$X_D[\Omega] \;=\; X[\Omega]-X_R[\Omega]. \tag{7}$$

We fit the shared transform $\tilde{T}$ by least squares over sampled rows from all query groups in the head,

$$\tilde{T} \;=\; \arg\min_{T}\big\|X_D[\Omega]-X_R[\Omega]\,T\big\|_F^2. \tag{8}$$

Applying the fitted transform to the retained-side outputs of all queries gives

$$\hat{X}_D \;=\; X_R\,\tilde{T} \;=\; [\,\hat{N}_D \mid \hat{Z}_D\,], \tag{9}$$

where $\hat{N}_D$ and $\hat{Z}_D$ are the predicted dropped-side value sums and normalization masses. We clamp $\hat{Z}_D$ to be nonnegative so that the total normalization mass remains positive. Proposition 1 then yields

$$\hat{o}_i \;=\; \frac{N_R[i]+\hat{N}_D[i]}{Z_R[i]+\hat{Z}_D[i]}. \tag{10}$$

Appendix C details the common-scale reconstruction of these unnormalized quantities, the row-sampling stride, and the numerical safeguards used by the least-squares solver.

Kernel implementation. Selection and compensation are implemented as GPU kernels. Selection computes the $\bar{q}$–key dot products, KEXACT reduction in equation 4, and value variance in equation 5 on chip without materializing token-level attention. Full attention is restricted to stride-s sampled query rows, reducing its arithmetic cost to a $1/s$ fraction, and runs on a separate CUDA stream alongside retained sparse attention. Batched FP64 Cholesky with iterative refinement solves the readouts, which are applied by batched matrix multiplication. Selection and compensation account for 0.31% and 2.59% of HunyuanVideo (Kong et al., 2025) generation time (2.90% combined), and 0.83% and 7.50% on Wan2.1 (Wan Team et al., 2025) (8.34% combined), versus 21.6% for SVG-EAR's full auxiliary pipeline on HunyuanVideo (Zhou et al., 2026).

3.3 Two-Dimensional Top-p Scheduling

A uniform top-p threshold does not account for differences in local attention error or its effect on final generation quality across denoising steps and layers. RDA therefore allocates retained density jointly across step–layer regions using calibrated local losses and sensitivities, subject to a fixed average-density budget. Figure 1(a) summarizes this offline procedure.

Six-region parameterization. We define six regions by combining three denoising-step segments—early, middle, and late—with two layer segments—shallow and deep. Each region uses one top-p threshold.

Table 1: Calibrated relative sensitivities g_Z used to weight regional attention loss on HunyuanVideo.

Relative sensitivity g_Z	Shallow layers (0–29)	Deep layers (30–59)
Early steps (6–13)	**7.2**	3.0
Middle steps (14–29)	3.6	1.5
Late steps (30–49)	2.4	1.0

Per-region coverage deficit. On a calibration trajectory, we sweep a grid of retained densities d. We define the regional loss $L_Z(d)$ as the mean relative squared error of the pre-compensation attention output:

$$L_Z(d) \;=\; \operatorname*{Avg}_{(t,l)\in Z,\; h} \frac{\left\|\hat{O}_{t,l,h}(d) - O_{t,l,h}\right\|_F^2}{\left\|O_{t,l,h}\right\|_F^2}, \tag{11}$$

where Z denotes a step–layer region and t, l, and h index denoising step, layer, and head. The sparse output $\hat{O}_{t,l,h}(d)$ retains the highest-scoring blocks at density d, and $O_{t,l,h}$ is the corresponding full-attention output. We refer to $L_Z(d)$ as the *coverage deficit*. We use this pre-compensation local distortion as a proxy for density allocation. On the same calibration data, we measure the mapping $D_Z(p)$ from each top-p threshold to its actual retained density. This mapping converts allocated densities into inference thresholds.

Two-dimensional sensitivity. Starting from a uniform sparse schedule, we restore full attention separately in each step or layer segment and use the resulting reductions in final generation error to fit relative sensitivities along the two axes. We represent region sensitivity g_Z as the product of its step and layer coefficients, normalized so that the late–deep region has sensitivity 1. The calibrated values in Table 1 assign greater sensitivity to early steps and shallow layers on HunyuanVideo.

Budget allocation by water filling. Under the assumption that regional influences on the final result are approximately additive, we choose one retained density per region using the surrogate objective

$$\min_{\{d_Z\}} \;\sum_Z w_Z\, g_Z\, L_Z(d_Z) \qquad \text{s.t.} \quad \sum_Z w_Z\, d_Z \;\le\; \bar{d}, \tag{12}$$

where w_Z is the fraction of sparse step–layer positions in region Z and $\bar{d}$ is the allowed average retained density over those positions. The objective, the *weighted coverage deficit*, combines regional losses using their calibrated sensitivities. It favors regions where additional density yields a larger sensitivity-weighted reduction in local distortion.

The measured density grid makes equation 12 a discrete allocation problem. We construct an allocation using greedy *water filling*. Starting from the sparsest candidate in every region, let d_Z^+ be the next denser candidate, with $\Delta d_Z = d_Z^+ - d_Z$ and $\Delta L_Z = L_Z(d_Z) - L_Z(d_Z^+)$. The gain per unit global density is

$$r_Z = \frac{w_Z g_Z\, \Delta L_Z}{w_Z\, \Delta d_Z} = g_Z \frac{\Delta L_Z}{\Delta d_Z}. \tag{13}$$

At each iteration, we increase the density of the feasible region with the largest r_Z, consuming $w_Z \Delta d_Z$ of the global budget. We stop when no further increment fits within $\bar{d}$. The allocated densities are then converted to thresholds p_Z through the measured mappings $D_Z(p)$. The resulting six thresholds form a lookup table indexed by denoising step and layer, with no online optimization.

4 Experiments

4.1 Settings

Models. We evaluate RDA on Wan2.1-T2V-14B (Wan Team et al., 2025) and HunyuanVideo (Kong et al., 2025). Both generate videos at 720×1280 resolution, with 81 frames for Wan2.1 and 129 frames for HunyuanVideo.

Baselines. We compare RDA with three representative training-free methods: SpargeAttn (Zhang et al., 2025a), SVG2 (Yang et al., 2025a), and SVG-EAR (Zhou et al., 2026). They cover uniform-block selection, content-based clustering, and clustering with compensation, respectively. Sparge

Table 2: Calibrated six-region top-p configurations targeting $\sim$80% sparsity. E, M, and L denote early, middle, and late step segments; S and D denote shallow and deep layer segments.

Model	Skip steps	E–S	E–D	M–S	M–D	L–S	L–D
Wan2.1-T2V-14B	12	0.85	0.22	0.66	0.069	0.164	0.055
HunyuanVideo	6	0.97	0.22	0.35	0.069	0.164	0.055

Table 3: End-to-end efficiency and quality on HunyuanVideo and Wan2.1. Speedup is measured against the corresponding FA3 dense baseline. Bold values indicate the best result among sparse methods. "–" indicates that FA3 serves as the full-attention reference.

Model	Method	Spar.	Speedup ↑	IQ ↑	DD ↑	MSE ↓	PSNR ↑	SSIM ↑	LPIPS ↓
HunyuanVideo	FA3	0	1.000×	67.2%	70.8%	–	–	–	–
	Sparge	49.9%	1.593×	34.5%	50.0%	0.0499	13.388	0.6908	0.6143
	SVG2	76.6%	1.837×	54.3%	23.3%	0.0062	22.268	0.8141	0.2862
	SVG-EAR	78.2%	1.571×	59.9%	58.3%	0.0107	20.280	0.8025	0.2720
	Ours	**78.3%**	**2.010×**	**62.1%**	**63.3%**	**0.0052**	**23.159**	**0.8444**	**0.2360**
Wan2.1	FA3	0	1.000×	67.3%	94.3%	–	–	–	–
	Sparge	49.9%	1.322×	62.5%	65.1%	0.0581	12.557	0.5435	0.4933
	SVG2	62.8%	1.576×	60.3%	65.3%	0.0229	16.564	0.6320	0.3994
	SVG-EAR	71.0%	1.620×	63.9%	60.2%	0.0386	15.001	0.5907	0.4102
	Ours	**77.1%**	**1.663×**	**68.7%**	**75.6%**	**0.0115**	**19.776**	**0.6440**	**0.3818**

denotes SpargeAttn in the tables. FlashAttention-3 (FA3) (Shah et al., 2024) serves as the exact dense reference.

Metrics. We measure fidelity to full-attention videos using MSE, PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). VBench's (Huang et al., 2024) imaging quality (IQ) and dynamic degree (DD) scores assess visual quality and motion, respectively.

Parameters. For both models, we use query groups and key blocks of size $B = 128$, a VAWARE weight of $w = 0.5$, and $\epsilon = 10^{-12}$ for numerical stability. We evaluate on 50 prompts selected across diverse styles from VBench. All experiments are conducted on NVIDIA H100 GPUs. Table 2 lists the calibrated six-region top-p schedules targeting approximately 80% sparsity. The "skip steps" column specifies the number of initial denoising steps that use full attention. Wan2.1 uses denoising-step segments 12–19, 20–49, and 50–99, crossed with layer segments 0–31 and 32–39. HunyuanVideo uses step segments 6–13, 14–29, and 30–49, crossed with layer segments 0–29 and 30–59.

4.2 End-to-End Results

Table 3 compares end-to-end efficiency and quality. Sparsity is the fraction of attention computation omitted across all attention calls, and speedup is measured relative to the corresponding FA3 baseline. The scheduling budget applies only to sparse step–layer positions, whereas the reported overall sparsity also accounts for the initial dense steps.

At the reported operating points, RDA achieves the highest speedup and the best fidelity among the evaluated sparse methods on both models. On HunyuanVideo, it reaches 2.010× speedup at 78.3% sparsity and leads the sparse baselines on all four fidelity metrics and both VBench scores. On Wan2.1, it reaches 1.663× speedup at 77.1% sparsity. Relative to SVG2, the strongest fidelity baseline on Wan2.1, RDA reduces MSE by 49.8% and improves PSNR by 3.212 dB.

4.3 Kernel Performance

With FA3 as the dense baseline, we compare the execution efficiency of the four sparse kernels at 30%, 70%, and 90% sparsity (Table 4).

Table 4: Kernel efficiency across sparsity levels on Wan2.1 and HunyuanVideo. Speedup is relative to dense FA3. Throughput counts retained QK and PV operations, excluding softmax; SVG-EAR timing includes centroid compensation and is therefore effective throughput. MFU normalizes throughput by the hardware peak at each kernel's arithmetic precision.

Model	Sparsity	Kernel	Time (ms) ↓	Speedup ↑	TFLOP/s ↑	MFU ↑
Wan2.1	30%	**Ours**	134.3	1.28×	611.1	**61.8%**
		SVG2	275.4	0.62×	297.5	30.1%
		SVG-EAR	290.5	0.59×	282.1	28.5%
		Sparge	117.3	1.46×	699.4	35.3%
	70%	**Ours**	58.3	2.94×	603.6	**61.0%**
		SVG2	109.5	1.57×	320.8	32.4%
		SVG-EAR	119.3	1.44×	294.3	29.7%
		Sparge	52.6	3.26×	670.8	33.9%
	90%	**Ours**	20.1	8.54×	583.2	**59.0%**
		SVG2	33.4	5.14×	350.1	35.4%
		SVG-EAR	48.1	3.57×	243.6	24.6%
		Sparge	20.1	8.54×	591.4	29.9%
HunyuanVideo	30%	**Ours**	207.0	1.28×	586.7	**59.3%**
		SVG2	363.3	0.73×	334.1	33.8%
		SVG-EAR	375.7	0.70×	323.1	32.7%
		Sparge	175.6	1.50×	695.3	35.1%
	70%	**Ours**	89.6	2.95×	581.0	**58.7%**
		SVG2	146.7	1.80×	354.6	35.8%
		SVG-EAR	161.8	1.63×	321.6	32.5%
		Sparge	77.8	3.39×	674.1	34.1%
	90%	**Ours**	30.5	8.66×	569.3	**57.5%**
		SVG2	44.1	5.99×	393.0	39.7%
		SVG-EAR	56.5	4.67×	307.2	31.1%
		Sparge	28.8	9.17×	611.7	30.9%

The FA3 reference latency is averaged over 4,000 Wan2.1 calls and 3,000 HunyuanVideo calls from dense profiles, yielding 171.6 ms and 264.1 ms, respectively. For each sparse kernel, we report the median latency over 20 isolated-kernel iterations on the same platform.

RDA's Hopper-oriented CuTe kernel, built on CUTLASS (Thakkar et al., 2023), achieves 57.5%–61.8% MFU versus 24.6%–39.7% for the baselines and outperforms SVG2 and SVG-EAR at all tested sparsities. Sparge (Zhang et al., 2025a) uses INT8 QK and FP8 PV, so its MFU is normalized by a hardware peak twice that of BF16, explaining its lower MFU despite competitive absolute speed. Table 3 shows that RDA provides the better overall speed–sparsity–fidelity trade-off at the reported operating points.

4.4 Sensitivity to Sparsity

Table 5 reports seven target sparsities on HunyuanVideo. Configurations t30–t80 use full attention for the first 6 denoising steps, while t90 uses 4 initial dense steps. PSNR changes little over t30–t50, then declines more sharply as sparsity increases. At 78.3% sparsity, RDA achieves 23.16 dB PSNR and 2.010× speedup.

4.5 Ablations

We compare three configurations across sparsity levels on HunyuanVideo in Figure 2. A0 is the uniform-density selection baseline without compensation. A1 uses KEXACT+VAWARE selection and linear-readout recovery, and A2 combines A1 with two-dimensional scheduling.

The combined selection-and-recovery module improves PSNR over A0 at every tested sparsity, and two-dimensional scheduling provides a further gain. Relative to A0, the full method improves PSNR by 11.716, 4.455, and 0.786 dB at target sparsities of 30%, 70%, and 90%, respectively. Appendix B separately examines block scoring and the interaction between partitioning and recovery.

Table 5: End-to-end speed and fidelity versus sparsity on HunyuanVideo.

Sparsity	Speedup	Gen. time (s)	PSNR	SSIM	LPIPS	MSE
30.2%	1.158×	989	25.57	0.898	0.113	0.0033
40.5%	1.273×	900	25.55	0.898	0.118	0.0033
50.8%	1.414×	810	25.45	0.895	0.126	0.0033
59.1%	1.557×	736	25.19	0.887	0.137	0.0035
69.1%	1.763×	650	24.34	0.873	0.171	0.0042
78.3%	2.010×	570	23.16	0.844	0.236	0.0052
89.2%	2.309×	496	18.63	0.776	0.356	0.0153

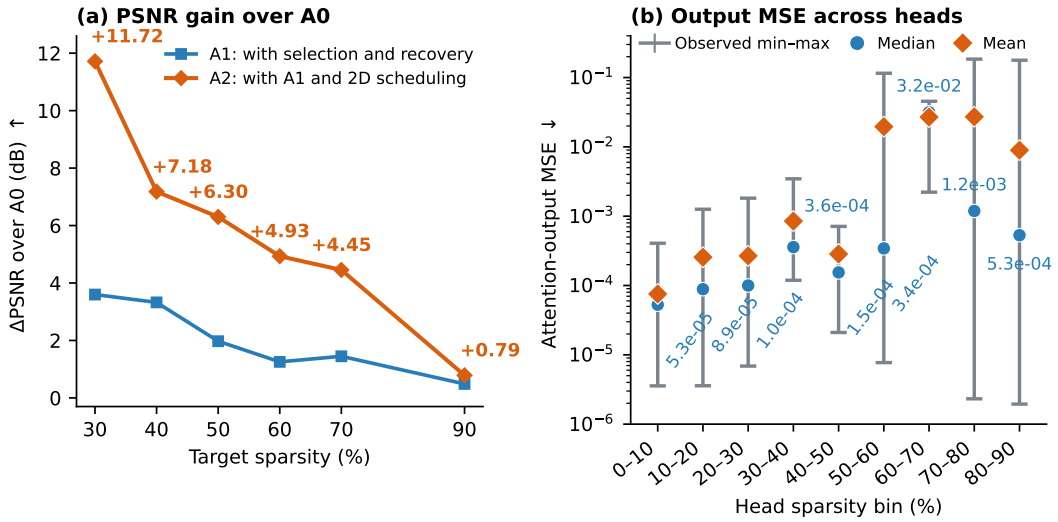

Figure 2: Quality ablation and attention-output error across sparsity levels. (a) PSNR gains at matched target sparsity: A0 is the uniform-density selection baseline without compensation; A1 uses KEXACT+VAWARE selection and linear-readout recovery; A2 combines A1 with two-dimensional scheduling. (b) Final attention-output MSE by head sparsity bin (log scale); bars span the minimum–maximum range, circles mark medians, and diamonds mark means.

Final attention-output MSE by head sparsity. Figure 2(b) groups post-recovery attention-output MSE by head sparsity. Median errors remain below 3.58×10^{-4} for bins under 60%. In the 60–70%, 70–80%, and 80–90% bins, the medians are 3.16×10^{-2}, 1.19×10^{-3}, and 5.32×10^{-4}, respectively. The gaps between means and medians, together with the min–max ranges, reveal substantial upper tails in several bins.

5 Conclusion

We presented RDA, a training-free method that predicts dropped attention contributions from retained augmented outputs through a per-head shared linear readout. Combined with recovery-oriented block selection and step–layer density scheduling, this approach reaches 78.3% and 77.1% sparsity on HunyuanVideo and Wan2.1, respectively. The corresponding end-to-end speedups over FA3 are 2.010× and 1.663×. RDA also achieves the best fidelity among the evaluated sparse methods on both models. These results demonstrate the value of reusing retained attention outputs for efficient sparse video generation.

REFERENCES

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, et al. Rethinking attention with performers, 2022. URL https://arxiv.org/abs/2009.14794.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Yizhao Gao, Zhichen Zeng, Dayou Du, et al. Seerattention: Learning intrinsic sparse attention in your llms, 2025. URL https://arxiv.org/abs/2410.13276.

Ziqi Huang, Yinan He, Jiashuo Yu, et al. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention, 2024. URL https://arxiv.org/abs/2407.02490.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. URL https://arxiv.org/abs/2001.04451.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, et al. Hunyuanvideo: A systematic framework for large video generative models, 2025. URL https://arxiv.org/abs/2412.03603.

Haopeng Li, Shitong Shao, Wenliang Zhong, et al. Pisa: Piecewise sparse attention is wiser for efficient diffusion transformers, 2026. URL https://arxiv.org/abs/2602.01077.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, et al. Moba: Mixture of block attention for long-context llms, 2025. URL https://arxiv.org/abs/2502.13189.

Xin Ma, Yaohui Wang, Xinyuan Chen, et al. Latte: Latent diffusion transformer for video generation, 2025. URL https://arxiv.org/abs/2401.03048.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Adam Polyak, Amit Zohar, Andrew Brown, et al. Movie gen: A cast of media foundation models, 2025. URL https://arxiv.org/abs/2410.13720.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers, 2020. URL https://arxiv.org/abs/2003.05997.

Jay Shah, Ganesh Bikshandi, Ying Zhang, et al. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, et al. CUTLASS, January 2023. URL https://github.com/NVIDIA/cutlass.

Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Wan Team, Ang Wang, Baole Ai, Bin Wen, et al. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Haocheng Xi, Shuo Yang, Yilong Zhao, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Shuo Yang, Haocheng Xi, Yilong Zhao, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025a.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025b. URL https://arxiv.org/abs/2408.06072.

Zihao Ye, Lequn Chen, Ruihang Lai, et al. Flashinfer: Efficient and customizable attention engine for llm inference serving. *arXiv preprint arXiv:2501.01005*, 2025. URL https://arxiv.org/abs/2501.01005.

Shai Yehezkel, Shahar Yadin, Noam Elata, et al. Accelerating text-to-video generation with calibrated sparse attention, 2026. URL https://arxiv.org/abs/2603.05503.

Jingyang Yuan, Huazuo Gao, Damai Dai, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, et al. Big bird: Transformers for longer sequences, 2021. URL https://arxiv.org/abs/2007.14062.

Jintao Zhang, Chendong Xiang, Haofeng Huang, et al. Spargeattn: Accurate sparse attention accelerating any model inference. In *International Conference on Machine Learning (ICML)*, 2025a.

Jintao Zhang, Kai Jiang, Chendong Xiang, et al. Spargeattention2: Trainable sparse attention via hybrid top-k+ top-p masking and distillation fine-tuning. *arXiv preprint arXiv:2602.13515*, 2026.

Peiyuan Zhang, Yongqi Chen, Runlong Su, et al. Fast video generation with sliding tile attention, 2025b. URL https://arxiv.org/abs/2502.04507.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

Xuanyi Zhou, Qiuyang Mang, Shuo Yang, et al. Svg-ear: Parameter-free linear compensation for sparse video generation via error-aware routing. *arXiv preprint arXiv:2603.08982*, 2026. Accepted to ECCV 2026.

A HELD-OUT VALIDATION OF THE SHARED READOUT

RDA pools sampled query rows from all query groups of a head to fit one transform $\tilde{T}$, which is then applied to every group. We directly test whether the fitted mapping transfers across query groups by comparing two held-out protocols across multiple HunyuanVideo generation trajectories, denoising steps, layers, and attention heads. Both protocols use the same retained-block selection, readout solver, and stride-16-equivalent fitting budget.

Evaluation metric. For held-out queries, we measure the *relative recovery error*

$$\rho_Q = \frac{\left\|X_D - \hat{X}_D\right\|_F^2}{\left\|X_D\right\|_F^2}, \tag{14}$$

where X_D is the ground-truth dropped-side augmented output and $\hat{X}_D = X_R\tilde{T}$ is its prediction. A zero prediction gives $\rho_Q = 1$, so the *capture* $1 - \rho_Q$ measures the fraction of squared error removed relative to this baseline. We compute ρ_Q separately over the held-out queries of each evaluated head unit.

H0: interleaved-row holdout. H0 excludes individual query rows from fitting while retaining fitting rows from the same query groups. It measures prediction on unseen queries under the standard row-level split.

H1: whole-group holdout. H1 removes all rows of selected 128-query groups before fitting $\tilde{T}$. The transform is learned exclusively from the remaining groups and then applied to the retained augmented outputs of the unseen groups. H1 therefore directly measures whether one fitted transform transfers across query groups.

Table 6 reports the median capture defined in Equation 14 and the mean reduction in normalized attention-output squared error relative to uncompensated sparse attention. Whole-group exclusion preserves the accuracy of the shared readout: median capture is 73.82% under H1, compared with 73.53% under H0, and the unit-level comparison favors H1 in the majority of cases.

Table 6: Held-out validation of sharing one readout across query groups. H1 removes complete 128-query groups from fitting, whereas H0 holds out individual rows.

Protocol	Capture ↑	Output reduction ↑
H0: interleaved rows	73.53%	43.20%
H1: whole query groups	**73.82%**	**44.40%**

The comparable H0 and H1 accuracy supports sharing one transform across query groups.

B ABLATIONS ON SELECTION AND RECOVERY

Each ablation fixes its own tensors, retained budget, and held-out query set, so comparisons are made within each experiment.

Block scoring. We fix the Q/K/V tensors collected from generation trajectories, the 128-token uniform blocks, the query split, and the shared-readout solver, varying only the scoring rule. Mean pooling uses the inner product of the mean query and mean key. KEXACT uses the log unnormalized attention mass of the mean query over all keys in the block. KEXACT+VAWARE adds the value-variance term with weight 0.5.

Table 7: Held-out ρ_Q under a matched retained budget, varying only the block-scoring rule.

Scoring	Held-out $\rho_Q \downarrow$
mean-pooling	0.225
KEXACT	0.185
KEXACT+VAWARE (RDA)	**0.183**

KEXACT reduces held-out ρ_Q from 0.225 with mean pooling to 0.185. Adding VAWARE provides a further reduction to 0.183, the lowest among the three scoring rules.

Partitioning and recovery. With tensors, held-out queries, and retained budget fixed, we evaluate all four combinations of uniform or k-means partitioning and shared linear readout or centroid recovery (Zhou et al., 2026).

Table 8: Held-out ρ_Q of the four combinations of partitioning and recovery under a matched retained budget.

	Shared linear readout	Centroid recovery
Uniform blocks	**0.293 (RDA)**	0.817
k-means clusters	0.451	0.517 (SVG-EAR)

With uniform blocks, shared linear readout reduces ρ_Q from 0.817 with centroid recovery to 0.293. The preferred partition depends on the recovery method in this comparison: shared linear readout performs better with uniform blocks than with k-means clusters, which give $\rho_Q = 0.451$, whereas centroid recovery performs better with k-means clusters. RDA's combination of uniform blocks and shared linear readout yields the lowest error among the four configurations.

C Numerical Details of Dropped-Output Recovery

Common exponential scale. The attention kernels return normalized outputs and natural-log normalizers, from which we reconstruct the unnormalized quantities. For a fixed head, let $o_{R,i}$ and $\ell_{R,i}$ denote the retained output and log-sum-exp for query i. The sampled full-attention call analogously returns o_i and ℓ_i for $i \in \Omega$. We choose a head-level shift $m = \max_i \ell_{R,i}$ over the retained queries in the recovery domain and reconstruct both sides on the same shifted scale:

$$Z'_{R,i} = \exp(\ell_{R,i} - m), \quad N'_{R,i} = o_{R,i} Z'_{R,i}, \qquad Z'_i = \exp(\ell_i - m), \quad N'_i = o_i Z'_i. \tag{15}$$

The dropped-side regression target for a sampled row is then

$$X'_{D,i} = [\, N'_i - N'_{R,i} \mid Z'_i - Z'_{R,i} \,], \qquad i \in \Omega. \tag{16}$$

Using the same m is essential because independently shifted full and retained outputs cannot be subtracted. The shift keeps every retained mass at most one and introduces only a common factor $\exp(-m)$ relative to Equation 1; this factor cancels in the final normalized output. After prediction, we clamp the predicted dropped mass $\hat{Z}'_D$ to be nonnegative and compute the compensated log-sum-exp as $m + \log(Z'_R + \hat{Z}'_D + 10^{-20})$.

Sampling used in the main results. We deterministically sample rows $\Omega = \{0, s, 2s, \ldots\}$ with stride $s = 16$ for both models. At the evaluated sequence lengths, this gives 4,725 sampled rows per head on Wan2.1 and 7,425 image-query rows per head on HunyuanVideo. The sampled full-attention call and the retained sparse call use the same queries, keys, values, and softmax scale. A fresh transform is fitted for every head within each recovery-enabled attention call and applied to all query rows; transforms are not reused across denoising steps or layers.

Regularized solve and rank-deficiency handling. Let $F = X'_R[\Omega]$ and $Y = X'_D[\Omega]$. Before solving, each column c of F is equilibrated by $s_c = \max(\|F_{:,c}\|_2, 10^{-30})$, giving $\bar{F} = F\,\mathrm{diag}(s^{-1})$. We solve

$$(\bar{F}^\top \bar{F} + \lambda I)U = \bar{F}^\top Y, \qquad \tilde{T} = \mathrm{diag}(s^{-1})U. \tag{17}$$

The Gram matrix is formed in FP32 and factorized by batched FP64 Cholesky. We use the fixed, non-tuned ladder $\lambda \in \{1, 10^3, 10^6\}$ and select the first value for which every factorization in the batch succeeds. Two iterative-refinement steps use the feature-space residual $Y - \bar{F}U$ to reduce the bias introduced by the diagonal shift. If all three Cholesky attempts fail, we fall back to `torch.linalg.lstsq`. If the solve itself raises an error, recovery is disabled for that call and the retained-only output is returned.

Output safeguards. We add a 10^{-20} stabilizer to divisions and logarithms involving reconstructed normalization masses. Rows with nonpositive reconstructed total mass fall back to the retained-only result. Finally, for each head, we compare the compensated and retained-only squared output errors on the sampled rows. Recovery is rejected for that head if this error ratio is non-finite or at least one. These checks reject unstable fits on the sampled rows.

D Offline Calibration of the Two-Dimensional Schedule

The two-dimensional schedule combines three denoising-step segments—early, middle, and late—with two layer segments—shallow and deep. Region boundaries and the six top-p thresholds are calibrated per model. At inference, the threshold is retrieved by denoising step and layer, with no online optimization.

Calibration curves. We collect Q/K/V from a representative generation trajectory at 4 denoising steps and 7 layers, covering 8 attention heads. At each of the 28 step–layer positions, we sweep 27 retained densities from 0.02 to 1.00 and measure the relative squared error of the pre-compensation sparse output against full attention. We aggregate these measurements into six regional loss curves $L_Z(d)$ and record each region's top-p-to-density mapping $D_Z(p)$ on the same data.

Sensitivity calibration. For each model, we perform seven complete generations: a full-attention reference, a uniform sparse baseline, and five restoration runs. Each restoration run uses full attention only in one segment—shallow layers, deep layers, early steps, middle steps, or late steps—and retains the sparse baseline elsewhere. Using the full-attention video as the reference, we measure each run's PSNR gain over the uniform sparse baseline. These gains, together with regional losses and sizes, determine the fitted step and layer sensitivities. Their products define the six coefficients g_Z, normalized so that the late–deep region has sensitivity one.

Budget allocation. Let w_Z denote the fraction of sparse step–layer positions in region Z. Given a target average density, we initialize all six regions at their sparsest candidates. For a candidate density increment Δd_Z that reduces regional loss by ΔL_Z, the marginal gain is $g_Z \Delta L_Z / \Delta d_Z$ and the global budget cost is $w_Z \Delta d_Z$. At each iteration, we apply the feasible increment with the largest marginal gain. We stop when no further increment fits, then convert the allocated densities to top-p thresholds using the measured mappings $D_Z(p)$.

Time cost. The complete offline calibration takes about 12 minutes per model on eight Hopper GPUs. This one-time cost is not counted toward per-video generation time.

E Measured Regional Calibration Curves

Figure 3 presents the measurements collected using the protocol in Appendix D. In panel (a), each point gives the regional mean relative squared error of the pre-compensation sparse output at a measured density. Loss decreases monotonically as density increases, but the curves differ across regions: equal density does not imply equal local attention error.

Panel (b) shows the actual retained density produced by each top-p threshold. The mapping is nonlinear and region-dependent, with density rising sharply as p approaches one. These differences motivate allocating the budget in density space and then converting each regional density to a top-p threshold.

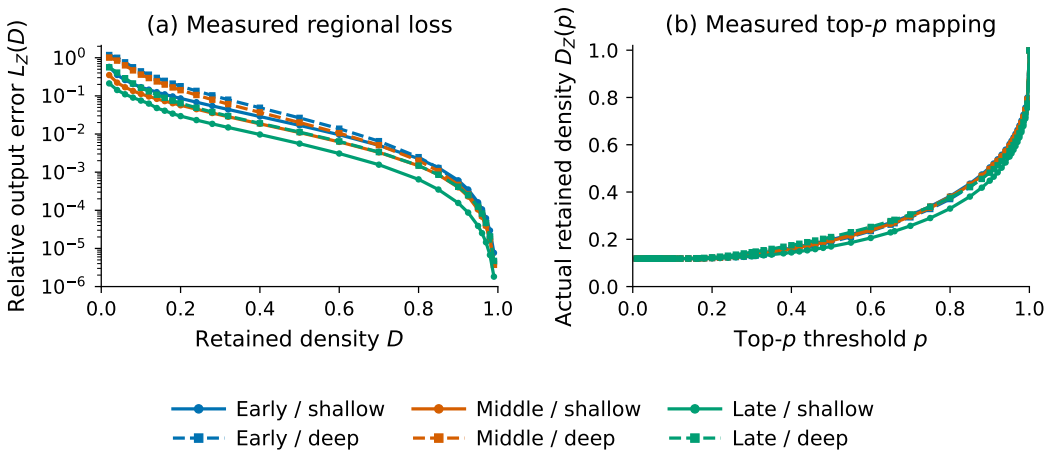

Figure 3: Measured curves for offline two-dimensional top-p calibration. (a) Regional pre-compensation block-sparse output error $L_Z(d)$ as a function of retained density. The full-density endpoint, whose error is numerically zero by construction, is omitted from the logarithmic axis. (b) Measured mapping $D_Z(p)$ from the top-p threshold to actual retained density for the approximately 80%-sparse configuration. Lines connect measured points without smoothing or curve fitting.

F Additional Results Across Sparsity Levels

Table 9: Quality and end-to-end speedup of sparse attention methods across target sparsity levels on HunyuanVideo. Bold values denote the best result at each sparsity level.

Target Sparsity	Method	PSNR ↑	Speedup ↑
30%	Sparge	13.96	**1.32**×
	SVG2	22.31	0.84×
	SVG-EAR	23.31	0.93×
	Ours	**25.57**	1.16×
40%	Sparge	13.88	**1.44**×
	SVG2	20.21	1.03×
	SVG-EAR	21.09	0.97×
	Ours	**25.55**	1.27×
50%	Sparge	13.39	**1.59**×
	SVG2	18.53	1.23×
	SVG-EAR	20.23	1.15×
	Ours	**25.45**	1.41×
60%	Sparge	13.49	**1.76**×
	SVG2	18.50	1.27×
	SVG-EAR	19.98	1.18×
	Ours	**25.19**	1.56×
70%	Sparge	13.31	**1.99**×
	SVG2	19.06	1.46×
	SVG-EAR	20.05	1.35×
	Ours	**24.34**	1.76×
80%	Sparge	12.98	**2.30**×
	SVG2	22.27	1.84×
	SVG-EAR	20.28	1.57×
	Ours	**23.16**	2.01×
90%	Sparge	12.81	**2.68**×
	SVG2	15.27	2.40×
	SVG-EAR	18.36	2.13×
	Ours	**18.63**	2.31×

RDA attains the highest PSNR at every target sparsity in Table 9. Over the 30%–80% range, its advantage over the best baseline at the same target ranges from 0.89 to 5.22 dB. Sparge achieves higher speedups at the same nominal sparsity, but its PSNR is 5.82–12.06 dB lower across the full sweep. Comparing operating points illustrates the resulting trade-off: RDA at 80% target sparsity exceeds Sparge at 60% by 9.67 dB while increasing speedup from 1.76× to 2.01×.

G End-to-End Latency Breakdown

Table 10: End-to-end latency breakdown of RDA on HunyuanVideo.

Component	Time (s)	Share
Attention	195.90	34.35%
Top-p block selection	1.78	0.31%
Output recovery	14.76	2.59%
Others	357.90	62.75%
Selection + recovery	**16.54**	**2.90%**
Total generation	570.33	100.00%

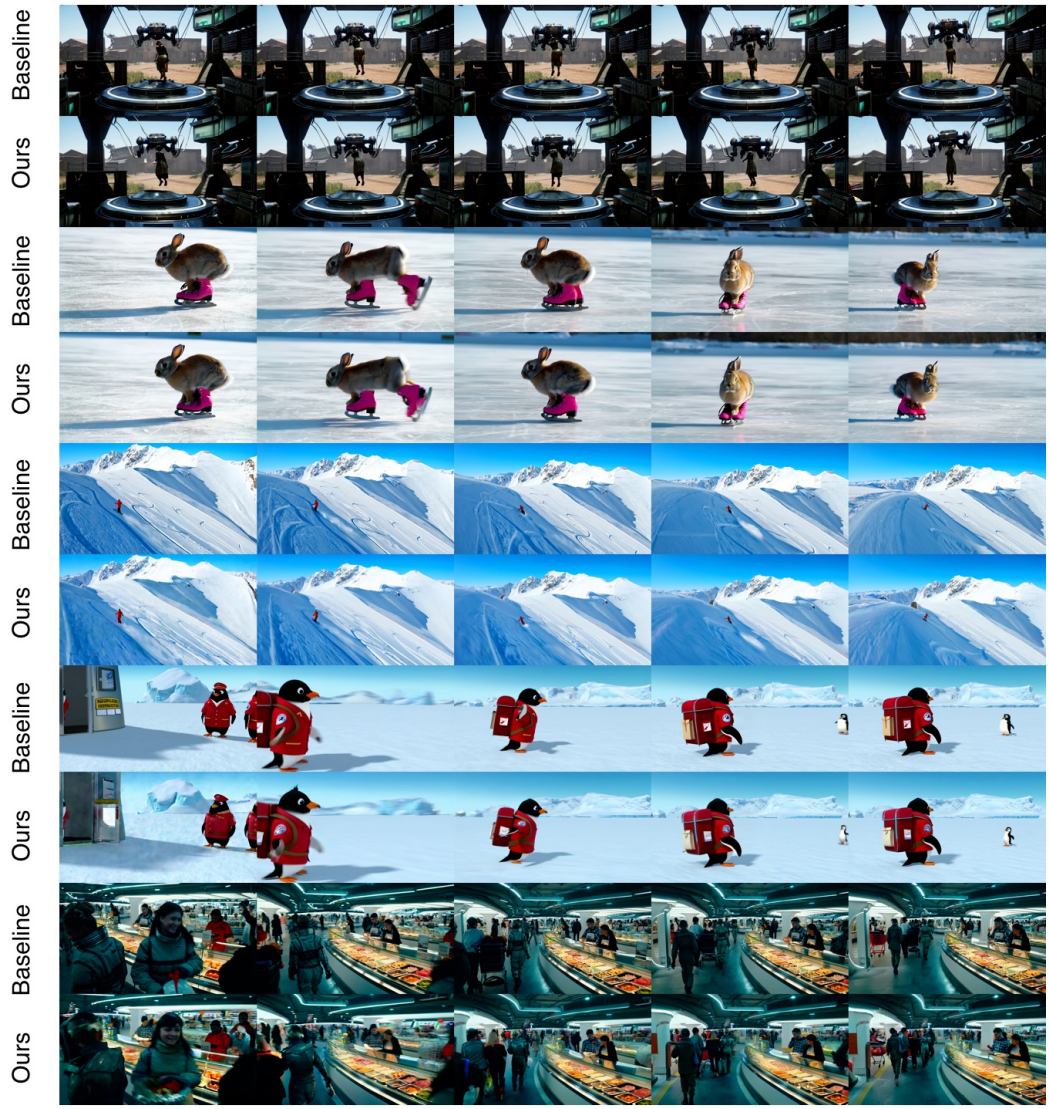

Figure 4: Qualitative result on Wan2.1 (example 1).

Selection and recovery together take 16.54 s, accounting for 2.90% of total generation time. Table 10 reports the individual components.

H Additional Visual Comparisons

Figures 4 and 5 present two qualitative examples from Wan2.1, and Figure 6 presents an example from HunyuanVideo.

I AI Assistance

AI-based writing assistance was used to improve the language and organization of the manuscript. The research ideas and all experimental work, including implementation, evaluation, and analysis, were completed by the authors.

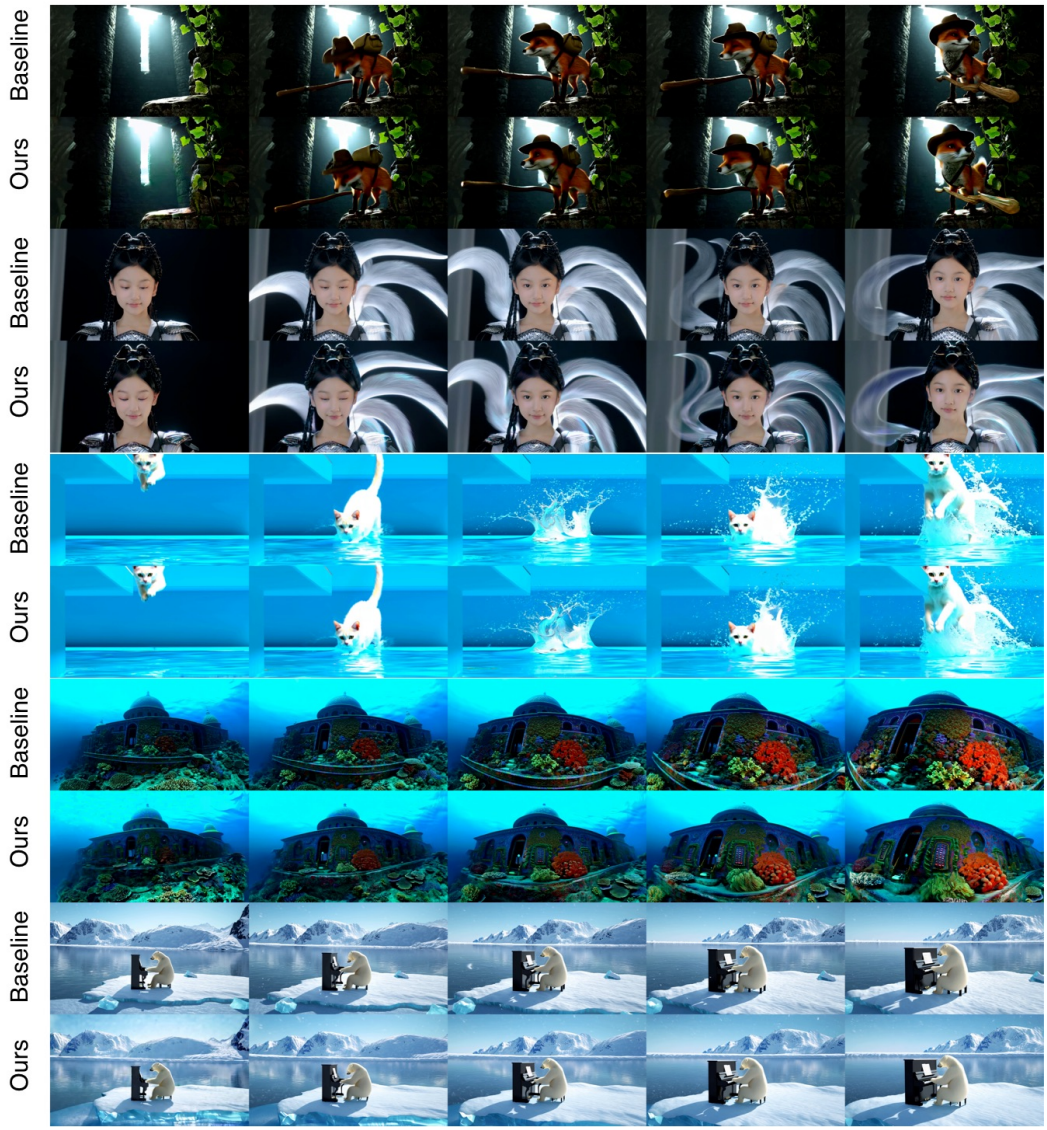

Figure 5: Qualitative result on Wan2.1 (example 2).

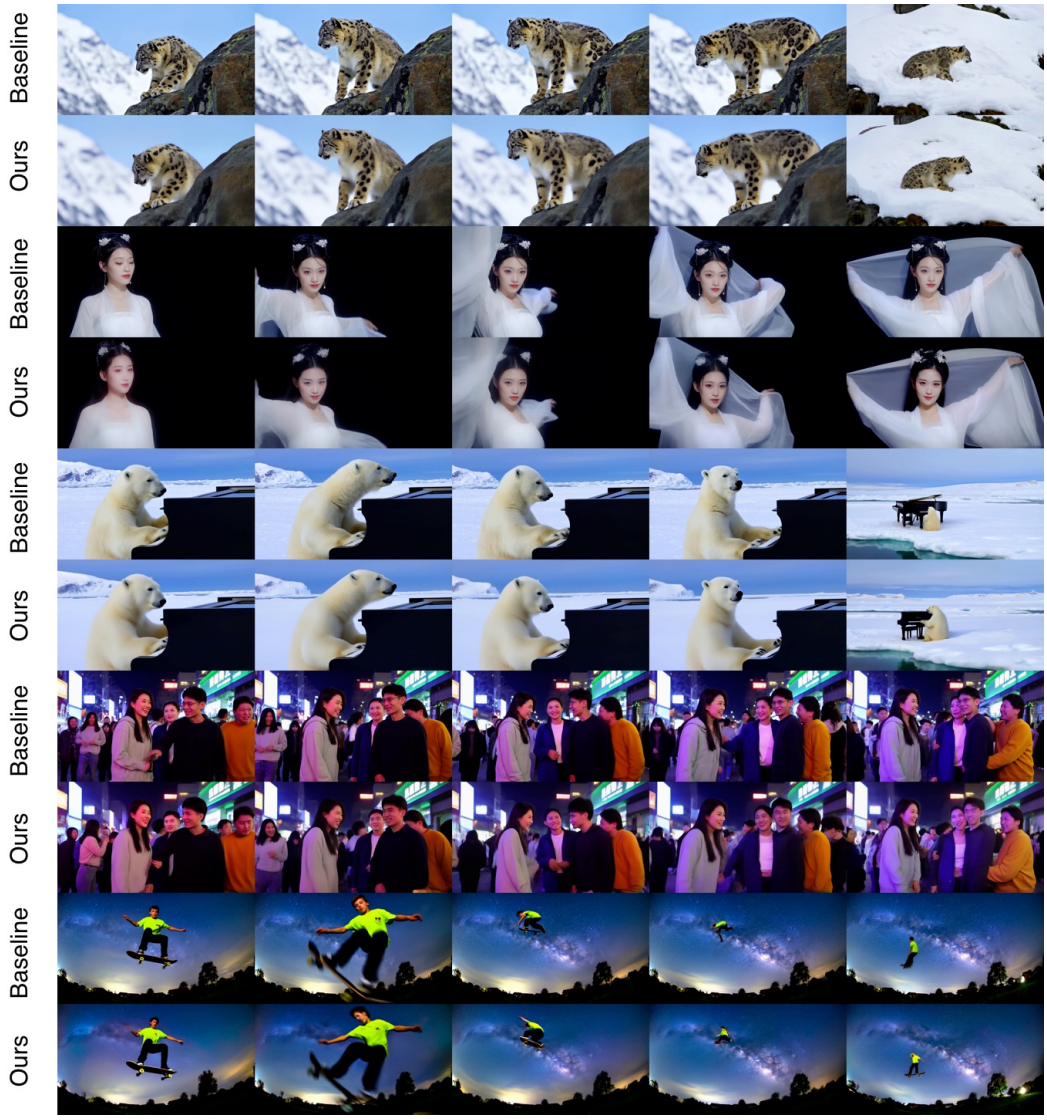

Figure 6: Qualitative result on HunyuanVideo.