EFFICIENT RESIDUAL-CENTRIC INFERENCE FOR SPACETIME AUTOREGRESSIVE VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Spacetime autoregressive video models combine multiscale spatial generation within each clip with temporal autoregression across clips, avoiding repeated full-resolution denoising while enabling previously generated visual states to be reused as context. This design, however, introduces two complementary inference bottlenecks: attention computation becomes concentrated at late, high-resolution scales, while reusable history creates a persistent KV cache. A natural solution is to sparsify attention and compress historical KV, but the two are typically optimized with different proxy objectives. We observe that once omitted attention is replaced by a cheap pooled correction, the relevant quantity is no longer raw attention importance or standalone tensor reconstruction error, but the *output residual left unexplained by the approximate attention operator*. Based on this observation, we introduce *residual-centric inference*, which provides a shared post-correction residual view for compute allocation and cache representation. We instantiate this principle with Output-Sensitive Routing (OSR), which allocates token-level attention using a proxy for post-correction residual risk, and Residual-Interaction Quantization (RIQ), which preserves pooled statistics and fits low-bit history representations to residual K–V interactions. On InfinityStar, OSR achieves up to $1.86\times$ end-to-end acceleration at native 720p/5s, while an orthogonal FFN FP8 optimization increases the complete-system speedup to $2.01\times$. At 480p/10s, OSR +RIQ achieves up to $1.50\times$ speedup and $5.74\times$ retained-history KV compression. Across both settings and tasks, these configurations largely preserve generation quality, with at most 0.31% relative degradation in VBench and VBench-I2V scores.

1 INTRODUCTION

Spacetime autoregressive video generation extends visual autoregression along two complementary axes: multiscale spatial refinement within each clip and temporal autoregression across clips. Within a clip, visual representations are generated from coarse to fine, with increasingly detailed grids predicted in parallel conditioned on earlier scales (Tian et al., 2024; Han et al., 2025). Across clips, previously generated visual states can be reused as context for subsequent generation. InfinityStar is a recent realization of this paradigm (Liu et al., 2025), combining next-scale spatial generation with cross-clip temporal context reuse (Figure 1(a)). By avoiding repeated full-resolution denoising, this paradigm provides a promising alternative to diffusion transformers for video generation (Peebles & Xie, 2023; Kong et al., 2024; Team Wan et al., 2025).

This spacetime structure incurs two inference bottlenecks. First, multiscale spatial refinement makes computation highly nonuniform: as resolution grows, attention cost becomes concentrated in a small number of late scales, where large current grids attend jointly to text and previously generated visual context. On InfinityStar, attention accounts for 68.6% and 70.8% of end-to-end latency for native 720p/5s and 480p/10s generation, respectively, and the final three scales of each video clip contribute 78.2% and 71.3% of total attention time (Figure 1(b)). Second, temporal autoregression introduces persistent state: the historical KV cache must remain resident and is repeatedly accessed throughout subsequent clip generation. At 480p/10s, live KV grows to 17.51 GiB as historical and current states coexist (Figure 1(c)). Efficient spacetime autoregressive inference therefore requires addressing both *late-scale attention computation* and *persistent historical state*.

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.

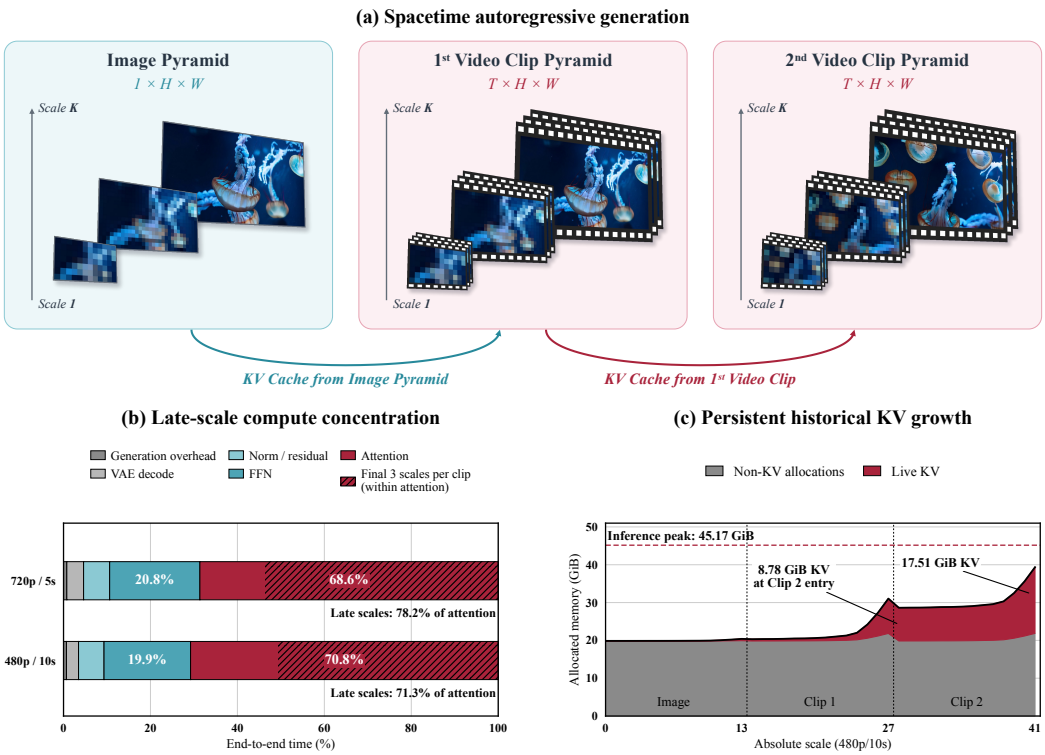


Figure 1: **Spacetime autoregressive video generation and its inference bottlenecks.** (a) Spacetime autoregression combines coarse-to-fine multiscale generation within each clip with visual-context reuse across clips. (b) Attention computation is concentrated at late high-resolution scales; hatched regions denote the final three scales of each clip. (c) Temporal context reuse introduces persistent historical KV, whose live memory grows as current states accumulate during subsequent clip generation. Together, spatial scale-up and temporal reuse create complementary compute and memory bottlenecks.

A natural response to these bottlenecks is to sparsify expensive attention and compress the history cache. Existing methods, however, generally optimize the two using different proxy objectives. Sparse attention identifies important interactions from attention scores, geometry, semantic structure, or learned routing (Zhang et al., 2025c; Xi et al., 2025; Yang et al., 2025; Zhang et al., 2025a;b), while KV quantization primarily seeks to preserve cached tensors under reduced precision (Liu et al., 2024; Hooper et al., 2024; Xi et al., 2026). Recent correction- and error-aware sparse methods explicitly account for approximation error (Li et al., 2026b;a; Zhou et al., 2026; Taghavi et al., 2026), but they remain focused on compute selection; cache representation is still largely optimized as a separate problem.

Our key observation is that *output correction changes the optimization target*. After pooled correction, attention mass alone does not determine a block's contribution to output error: a high-mass block may be accurately approximated, whereas a lower-mass block may retain a substantial residual. Compute allocation should therefore account for what the correction leaves unexplained, rather than only how much attention a block receives. The same principle applies to cache compression. Small elementwise KV reconstruction error need not preserve the block statistics used by pooled correction or the residual K–V interactions beyond those statistics. Quantization should thus preserve the information needed by the corrected attention operator.

To characterize the remaining error, we write

$$O_{\text{dense}} = O_{\text{approx}} + \Delta O_{\text{res}}, \tag{1}$$

where O_{approx} combines exact contributions from selected interactions with pooled correction for omitted blocks under a shared softmax normalization. The post-correction output residual ΔO_{res} measures the discrepancy that remains after both contributions are included. It therefore provides

a common reference for assessing which interactions require token-level computation and which information compressed history should retain.

Figure 2 illustrates the resulting *residual-centric inference* framework. We instantiate this view with two complementary mechanisms: Output-Sensitive Routing (OSR) uses compact block and query statistics to estimate residual risk for compute allocation, while Residual-Interaction Quantization (RIQ) encodes reusable history to support both pooled correction and token-level attention. We evaluate these components on a spacetime autoregressive video generation model, targeting late-scale computation and persistent history, respectively.

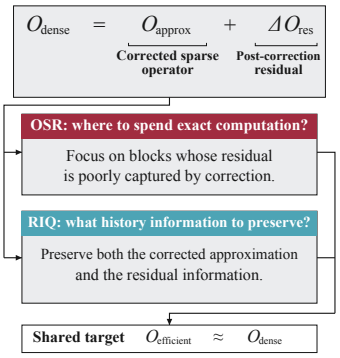


Figure 2: **Residual-centric inference.** OSR allocates token-level computation to blocks with large predicted post-correction residuals; RIQ preserves pooled statistics and residual K–V interactions in compressed history.

Our contributions are:

• We formulate corrected sparse attention through its **post-correction output residual**, providing a shared residual-based design perspective for compute allocation and persistent-cache representation.

• Guided by this residual view, we develop two training-free mechanisms. **OSR** estimates post-correction routing risk from compact block and query statistics, while **RIQ** preserves pooled statistics and residual K–V interactions under INT2 history-cache compression.

• We evaluate the proposed inference stack for spacetime autoregressive video generation, using InfinityStar as the experimental backbone. At native 720p/5s, OSR achieves $1.86\times$/$1.82\times$ end-to-end speedup for text-to-video (T2V)/image-to-video (I2V), increasing to $2.01\times$/$1.99\times$ with orthogonal FFN FP8 optimization. At 480p/10s, OSR +RIQ achieves $1.50\times$/$1.47\times$ end-to-end speedup for T2V/I2V and $5.74\times$ retained-history KV compression. Aggregate VBench and VBench-I2V scores remain close to dense references across both regimes.

2 Related Work

Efficient visual autoregressive generation. For image generation, VAR and Infinity predict finer spatial grids in parallel conditioned on coarser scales (Tian et al., 2024; Han et al., 2025), in contrast to iterative diffusion denoising (Peebles & Xie, 2023; Kong et al., 2024; Team Wan et al., 2025) and token-wise autoregression (Sun et al., 2024). Acceleration methods for multiscale autoregression exploit token and scale redundancy (Guo et al., 2025; Mao et al., 2026), reduce attention cost through sparsity and lower-precision computation (Xie et al., 2024), and adapt cache allocation across scales (Li et al., 2025a). For video generation, InfinityStar extends this multiscale paradigm through cross-clip visual-context reuse (Liu et al., 2025), while FastSTAR accelerates this setting through token pruning and partial refinement updates (Yune et al., 2026). We target the same spacetime autoregressive setting, preserving the native token grid and generation schedule while combining corrected sparse attention with compression of reusable history.

Sparse attention with output correction. Video sparse attention reduces computation through geometric masks (Zhang et al., 2025c; Li et al., 2025b), content-adaptive routing (Xi et al., 2025; Zhang et al., 2025a; Yang et al., 2025), or learned tile selection (Zhang et al., 2025b). Beyond selection alone, PISA and Sol-Attn approximate omitted contributions through Taylor- or proxy-based correction (Li et al., 2026b;a), while SVG-EAR and SparsePR incorporate output-error information through denser probes or residual fitting (Zhou et al., 2026; Taghavi et al., 2026). These works establish the importance of approximation-aware sparse attention. OSR instead derives its routing proxy from the post-correction residual of the executed approximate operator: compact block moments and query statistics estimate residual risk after pooled correction, without semantic regrouping or dense-output probes.

KV cache quantization. Language-model KV quantizers address asymmetric K/V statistics and outliers (Liu et al., 2024; Hooper et al., 2024), redistribute activation ranges through rotations (Ashk-

boos et al., 2024; Su et al., 2025), or combine mean-centering with layer-adaptive precision (Joshi et al., 2025). For video generation, QVG exploits spatiotemporal redundancy through semantic-aware smoothing and progressive residual quantization (Xi et al., 2026), while related work corrects quantization-induced softmax bias between compressed history and unquantized current tokens (Tuncer et al., 2026). RIQ accounts for pooled correction by retaining block statistics and fitting key scales to preserve the centered K–V interaction that governs the leading content residual.

3 METHOD

3.1 ATTENTION IN SPACETIME AUTOREGRESSIVE GENERATION

In the spacetime autoregressive models considered here, attention follows a spacetime sparse attention (SSA) pattern (Liu et al., 2025). At each spatial scale, queries attend to text ($\mathcal{I}_T$), reusable history ($\mathcal{I}_H$), and current-scale visual tokens ($\mathcal{I}_C$). As illustrated in Figure 3, history retains the preceding clip's final-scale visual KV, while other historical scales are masked. The first clip conditions on image history; subsequent clips reuse video history. S1–S3 denote increasing spatial scales.

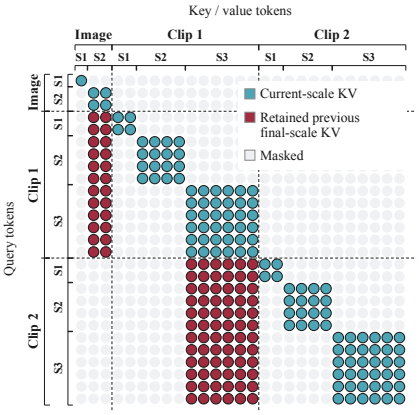


Figure 3: **Native visibility in SSA.**

All visible sources share a single softmax normalization. With head dimension d and scaled query $q = Q/\sqrt{d}$, the attention output is

$$O_{\text{dense}}(q) = \frac{\sum_{u\in\{T,H,C\}} \sum_{i\in\mathcal{I}_u} e^{q^\top K_i} V_i}{\sum_{u\in\{T,H,C\}} \sum_{i\in\mathcal{I}_u} e^{q^\top K_i}}. \quad (2)$$

Throughout, O_{dense} denotes exact attention over all tokens visible under the SSA mask.

3.2 RESIDUAL DECOMPOSITION OF CORRECTED SPARSE ATTENTION

We first isolate sparsification error assuming unquantized KV: selected blocks use their original tokens, and omitted blocks receive pooled correction. Section 3.4 addresses the additional error from quantizing history. Partition visual KV into blocks of up to B valid tokens. For a block b of size $n_b \le B$, let $Z_b = \sum_{i\in b} e^{q^\top K_i}$ be its softmax mass and $N_b = \sum_{i\in b} e^{q^\top K_i} V_i$ its value-weighted sum. With block means $\bar{K}_b$ and $\bar{V}_b$, zeroth-order pooled correction (Li et al., 2026a) uses

$$\widehat{Z}_b = n_b e^{q^\top \bar{K}_b}, \qquad \widehat{N}_b = \widehat{Z}_b \bar{V}_b. \quad (3)$$

Let $\mathcal{U}$ denote the omitted visual blocks. Let N_{exact} and Z_{exact} denote the numerator and normalization contributions from selected visual blocks and non-sparsified tokens. The corrected sparse operator is

$$O_{\text{approx}} = \frac{N_{\text{approx}}}{Z_{\text{approx}}} = \frac{N_{\text{exact}} + \sum_{b\in\mathcal{U}} \widehat{N}_b}{Z_{\text{exact}} + \sum_{b\in\mathcal{U}} \widehat{Z}_b}. \quad (4)$$

Token-level and pooled contributions thus share a single softmax normalizer.

Define the omitted blocks' numerator and mass errors as $\delta N = \sum_{b\in\mathcal{U}}(N_b - \widehat{N}_b)$ and $\delta Z = \sum_{b\in\mathcal{U}}(Z_b - \widehat{Z}_b)$, respectively. Because $N_{\text{dense}} = N_{\text{approx}} + \delta N$ and $Z_{\text{dense}} = Z_{\text{approx}} + \delta Z$, subtracting the two normalized outputs gives the exact identity

$$O_{\text{dense}} = O_{\text{approx}} + \underbrace{\frac{\delta N - O_{\text{approx}}\delta Z}{Z_{\text{approx}} + \delta Z}}_{\Delta O_{\text{res}}}. \quad (5)$$

With unquantized selected KV, ΔO_{res} is precisely the error left by pooling. To characterize it, center each omitted block: $dK_i = K_i - \bar{K}_b$, $dV_i = V_i - \bar{V}_b$, and $x_i = q^\top dK_i$. Let $\mathbb{E}_b$ denote the mean over its n_b valid tokens. Then $\mathbb{E}_b[dK_i] = \mathbb{E}_b[dV_i] = 0$. Define $h_b = \mathbb{E}_b[(e^{x_i} - 1)dV_i]$

and $m_b = \mathbb{E}_b[e^{x_i}] - 1$. The former measures value residuals correlated with key logits; the latter measures excess softmax mass. The block errors satisfy $\delta N_b = \widehat{Z}_b(h_b + m_b\bar{V}_b)$ and $\delta Z_b = \widehat{Z}_b m_b$. Substituting these into Equation 5 exposes each omitted block's contribution to output error:

$$\Delta O_{\text{res}} = \frac{\sum_{b\in\mathcal{U}} \widehat{Z}_b \left[h_b + m_b(\bar{V}_b - O_{\text{approx}})\right]}{Z_{\text{approx}} + \sum_{b\in\mathcal{U}} \widehat{Z}_b m_b}. \tag{6}$$

Let $R_K, R_V \in \mathbb{R}^{n_b\times d}$ stack the centered residuals. When within-block centered logits x_i are small, expanding e^{x_i} gives the leading terms:

$$h_b \approx \frac{1}{n_b} R_V^\top R_K q, \qquad m_b \approx \frac{1}{2n_b} q^\top R_K^\top R_K q. \tag{7}$$

The first-order term in m_b vanishes because the keys are centered. Thus, the leading content term depends on the K–V cross-moment $R_K^\top R_V$, and the leading mass term depends on key covariance $R_K^\top R_K$. This Taylor approximation requires small centered logits; Equation 6 itself is exact.

These motivate two components (Figure 4). Output-sensitive routing (OSR) estimates the query-dependent error of pooling each block. Residual-interaction quantization (RIQ) stores the means required for correction and compresses historical residuals with an interaction-aware objective.

3.3 Output-Sensitive Routing

OSR routes visual blocks by the estimated effect of replacing their tokens with pooled correction. It evaluates high-risk blocks at token resolution and pools the rest.

Low-rank residual statistics. The moments in Equation 7 are $d \times d$ matrices. OSR instead summarizes each block along one direction of key variation. Starting from $w_0 = \text{unit}(dK_1)$, one power iteration and two projections give

$$W_b = \text{unit}\big(\mathbb{E}_b[dK_i(dK_i^\top w_0)]\big), \quad U_b = \mathbb{E}_b[dV_i t_i], \quad \lambda_b = \mathbb{E}_b[t_i^2], \tag{8}$$

where $t_i = dK_i^\top W_b$ and $\text{unit}(x) = x/\sqrt{\|x\|_2^2 + \epsilon}$. W_b estimates a leading key direction, λ_b its projected variance, and U_b the associated K–V cross-moment. Along this direction, $W_b U_b^\top$ approximates $R_K^\top R_V/n_b$ and $\lambda_b W_b W_b^\top$ approximates $R_K^\top R_K/n_b$.

For a query block with mean $\bar{q}$ and diagonal variance σ_q^2, we estimate the second moment of $q^\top W_b$ as $r_b^2 = (\bar{q}^\top W_b)^2 + \langle\sigma_q^2, W_b^2\rangle$. This uses a diagonal approximation to the query covariance.

Correction-aware omission risk. Equation 6 also weights each block by its softmax mass. For a context domain $\mathcal{D}$, we approximate it using block statistics. Set $\ell_b = \log n_b + \bar{q}^\top \bar{K}_b$ and $v_b = \sqrt{\langle\sigma_q^2, \bar{K}_b^2\rangle}$, and compute

$$O_0^{\mathcal{D}} = \frac{\sum_{j\in\mathcal{D}} e^{\ell_j}\bar{V}_j}{\sum_{j\in\mathcal{D}} e^{\ell_j}}, \qquad \bar{p}_b^{\mathcal{D}} = \min\left\{1, \frac{e^{\ell_b + \kappa v_b}}{\sum_{j\in\mathcal{D}} e^{\ell_j - \kappa v_j}}\right\}, \tag{9}$$

with $\kappa = 0.5$. $O_0^{\mathcal{D}}$ anchors the output scale. To compute $\bar{p}_b^{\mathcal{D}}$, we shift the numerator logit upward by κv_b and each denominator logit downward by κv_j. These heuristic weights need not sum to one. The global domain covers visible context, while source domains contain eligible history or current-context blocks.

Approximate the content and excess-mass terms, respectively, by

$$A_b^{\mathcal{D}} = \bar{p}_b^{\mathcal{D}} \frac{r_b\|U_b\|_2}{\|O_0^{\mathcal{D}}\|_2 + \epsilon}, \qquad B_b^{\mathcal{D}} = \frac{1}{2}\bar{p}_b^{\mathcal{D}} r_b^2 \lambda_b.$$

We combine them into a routing cost:

$$\widetilde{\rho}_b^{\mathcal{D}}(\tau) = \frac{A_b^{\mathcal{D}}}{\tau} + B_b^{\mathcal{D}} + \gamma\bar{p}_b^{\mathcal{D}}. \tag{10}$$

$A_b^{\mathcal{D}}$ estimates content error; $B_b^{\mathcal{D}}$ tracks excess mass but omits its output-dependent factor $(\bar{V}_b - O_{\text{approx}})$. The last term guards high-mass blocks. The parameter $\tau > 0$ rescales the content term and can therefore change the block ranking. This cost is a routing heuristic, not a certified output-error bound.

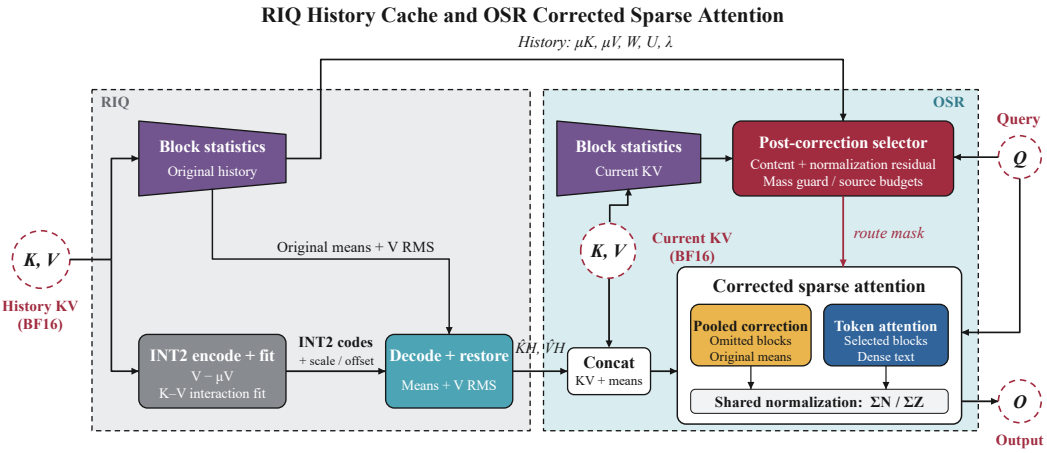


Figure 4: **Operator realization of residual-centric inference.** OSR uses query and block statistics to route visual blocks between token-level attention and pooled correction, while RIQ stores pooled statistics together with compressed history residuals. Selected and corrected contributions share a single softmax normalization, implementing the compute- and memory-side objectives derived in Section 3.2.

Source-aware budgeting. Global normalization can underweight blocks that are important within a smaller source. We score each visual block both globally and within its source $s(b) \in \{H, C\}$. We take the larger score and pool low-score blocks under a unit budget:

$$\widetilde{c}_b = \max\{\widetilde{\rho}_b^{\text{global}}(\tau), \widetilde{\rho}_b^{s(b)}(\tau)\}, \qquad \sum_{b \in \mathcal{U}} \widetilde{c}_b \le 1.$$

Mandatory-retention rules apply after selection. A histogram approximates this selection without a full sort, so density adapts per head and query block. The unit budget is measured in proxy-score units and has no formal error guarantee.

Corrected sparse output. Selected blocks enter token-level attention; omitted blocks contribute Equation 3. Text and protected tokens share the normalizer in Equation 4. With RIQ, selected historical tokens use reconstructed KV, so the implemented output also includes quantization error. Block moments affect routing only; correction remains zeroth order. CFG branches share routes and query statistics, while native visibility and GQA are preserved (Algorithm 1 in Appendix A.1).

3.4 Residual-Interaction Quantization

RIQ compresses reusable history. The blockwise quantities h_b and m_b also characterize selected blocks' token-level contributions beyond pooling. Guided by their leading terms in Equation 7, we retain original means for pooled correction, quantize value residuals, and fit key scales to reduce K–V interaction distortion while regularizing key reconstruction error.

Operator-aligned mean–residual representation. For each history block, write $K = \mathbf{1}\mu_K^\top + R_K$ and $V = \mathbf{1}\mu_V^\top + R_V$, where μ_K, μ_V are original block means and R_K, R_V are centered residuals. RIQ stores the means and OSR summaries explicitly and encodes residuals in INT2 (Figure 4). Pooled history blocks use the original means; selected ones decode token residuals. Current KV, text, and incomplete or protected history tails remain BF16. Only complete B-token history blocks are quantized.

Value residual quantization and restoration. For values, we fit 2-bit affine codes per token and 64-channel group with three assignment/fitting iterations. Let $\widetilde{R}_V$ be the decoded residual. Remove its blockwise channel mean and restore its original root-mean-square (RMS) magnitude:

$$\widehat{V}_i = \mu_V + \frac{r_V}{\sqrt{\widetilde{r}_V^2 + \epsilon}} \left(\widetilde{R}_{V,i} - \text{mean}_j \, \widetilde{R}_{V,j} \right). \tag{11}$$

Here r_V and $\widetilde{r}_V$ are the RMS values of R_V and the centered decoded residual, respectively; $\widehat{R}_V$ denotes the restored centered residual. Means use BF16; RMS and affine parameters use FP16.

Interaction-aware key fitting. The leading content term would be preserved if reconstructed residuals satisfied $\widehat{R}_K^\top \widehat{R}_V \approx R_K^\top R_V$. We fit keys against original R_V to isolate key-induced distortion. This surrogate does not capture all error from quantized values; value restoration above addresses their mean and magnitude separately.

First quantize each key channel over B tokens with 2-bit affine codes. Fix integer codes $C \in \{0,1,2,3\}^{B\times d}$ and center each column as $T = C - \mathbf{1}\,\mathrm{mean}_i(C_i)^\top$. With $\widehat{R}_K(s) = T\,\mathrm{diag}(s)$ and $E_K(s) = \widehat{R}_K(s) - R_K$, we fit

$$\min_s \ \frac{\|E_K(s)^\top R_V\|_F^2}{\max(\|R_V\|_F^2/B, \epsilon)} + \lambda \|E_K(s)\|_F^2. \tag{12}$$

The first term penalizes key-induced interaction error; the second limits key reconstruction error, including changes in key dispersion. With codes fixed, this quadratic objective separates by key channel. Define $M = R_V R_V^\top / \max(\|R_V\|_F^2/B, \epsilon) + \lambda I$. For channel c, the fitted quantization scale is

$$s_c = \mathrm{clip}\left(\frac{t_c^\top M r_{K,c}}{\max(t_c^\top M t_c, \epsilon)},\ 0.5 s_{\mathrm{old},c},\ 1.5 s_{\mathrm{old},c}\right). \tag{13}$$

Here t_c and $r_{K,c}$ are column c of T and R_K; $s_{\mathrm{old},c}$ is the initial reconstruction scale. Clipping limits deviation from that scale. We use $B = 64$ and set $\lambda = 1$ based on the scan in Appendix C.4. Fitting occurs once when history enters the cache, without future queries or dense outputs. Only packed INT2 codes and compact metadata persist. The same codes support reconstruction-oriented scales for dense reads and interaction-fitted scales for corrected sparse reads. Appendix A.2 provides the encoding and read procedures (Algorithm 2) and the affine-fitting subroutine (Algorithm 3).

4 Experiments

4.1 Experimental Setup

Workloads and baselines. We evaluate InfinityStar (Liu et al., 2025) on T2V and I2V at native 720p/5s and 480p/10s, covering high-resolution generation and longer-duration generation with cross-clip KV reuse. All experiments use a single NVIDIA H200 and official image–text pairs for I2V. At 720p/5s, we compare Dense, PISA (Li et al., 2026b), Sol-Attn (Li et al., 2026a), SVG2 (Yang et al., 2025), FastSTAR (Yune et al., 2026), and OSR with or without FFN FP8. At 480p/10s, we compare Dense and OSR combined with KIVI INT2 (Liu et al., 2024), QVG INT2 (Xi et al., 2026), or RIQ. Dense is the native InfinityStar baseline; all accelerated methods share common system optimizations. Implementation details and a dense-attention control isolating these optimizations appear in Appendices B and C.1, respectively. We use $\tau = 1.6$, $\gamma = 0.3$, $\kappa = 0.5$, and 64-token blocks, with parameter sensitivity reported in Appendix C.3. OSR sparsifies scales 27–29 at 720p/5s and scales 27 and 39–41 at 480p/10s, corresponding to Clip 1's final scale and Clip 2's final three scales.

Metrics. We measure framewise fidelity to Dense using PSNR, SSIM (Wang et al., 2004), and AlexNet-based LPIPS (Zhang et al., 2018), and generation quality using VBench (Huang et al., 2024) for T2V and VBench-I2V (Huang et al., 2026) for I2V on their official evaluation sets with five seeds. Efficiency is measured by end-to-end latency and global attention density. Appendices C.7 and B.1 detail benchmark score decomposition and uncertainty, and efficiency measurement, respectively.

4.2 Main Results

We evaluate attention acceleration at 720p/5s and joint attention acceleration and history compression at 480p/10s.

High-resolution generation (720p/5s). OSR alone achieves $1.86\times$/$1.82\times$ end-to-end speedup for T2V/I2V (Table 1). Orthogonal FFN FP8 optimization increases this to $2.01\times$/$1.99\times$, with 32.04/29.99 dB PSNR. Restricting OSR to current visual KV, matching the sparse-source scope of the diffusion-derived baselines, still yields $1.85\times$ T2V speedup (Appendix C.2). Thus, its acceleration in this setting does not primarily depend on sparsifying history.

Table 1: **Quantitative comparison of T2V and I2V quality and efficiency at 720p/5s.** †FastSTAR is our reimplementation; Appendix C.6 compares end-to-end deployment configurations across operating points.

Method	Time (s)↓	Speedup	Global density (%)	PSNR (dB)↑	SSIM↑	LPIPS↓	VB↑
T2V							
Dense	56.49	1.00×	100.00	∞	1.0000	0.0000	84.76
PISA	33.64	1.68×	23.30	30.36	0.8817	0.0696	84.09
Sol-Attn	33.34	1.69×	23.95	29.80	0.8714	0.0771	83.99
SVG2	81.80	0.69×	27.73	30.88	0.8841	0.0678	84.03
FastSTAR†	33.66	1.68×	41.32	28.03	0.8368	0.1146	83.69
OSR	30.35	1.86×	19.45	**32.31**	**0.9056**	**0.0537**	**84.82**
OSR + FP8	**28.08**	**2.01×**	19.46	32.04	0.9016	0.0557	84.53
I2V							
Dense	56.59	1.00×	100.00	∞	1.0000	0.0000	88.45
PISA	33.41	1.69×	20.02	24.92	0.7926	0.1394	87.09
Sol-Attn	33.62	1.68×	23.57	27.80	0.8584	0.0849	88.05
SVG2	74.94	0.76×	27.53	28.40	0.8660	0.0797	88.04
FastSTAR†	34.87	1.62×	41.20	25.45	0.8086	0.1313	87.48
OSR	31.11	1.82×	19.08	**30.30**	**0.8972**	**0.0581**	**88.41**
OSR + FP8	**28.50**	**1.99×**	19.08	29.99	0.8926	0.0606	88.18

Longer-duration generation (480p/10s). OSR with RIQ achieves 1.50×/1.47× end-to-end speedup for T2V/I2V and 5.74× retained-history KV compression, with 27.25/28.13 dB PSNR (Table 2). Aggregate VBench and VBench-I2V scores remain close to their dense references. Appendix C.1 distinguishes persistent history storage from peak inference allocated memory; Appendix C.5 separates the effects of attention routing and cache representation.

Table 2: **Quantitative comparison of T2V and I2V quality and efficiency at 480p/10s.** KV compression is measured relative to OSR's BF16 history representation, including shared routing statistics.

Method	Time (s)↓	Speedup	Global density (%)	KV compression ratio↑	PSNR (dB)↑	SSIM↑	LPIPS↓	VB↑
T2V								
Dense	69.89	1.00×	100.00	1.00×	∞	1.0000	0.0000	81.89
OSR + KIVI	56.50	1.24×	40.86	4.86×	25.08	0.7148	0.2096	80.97
OSR + QVG	52.26	1.34×	40.38	**6.14×**	24.79	0.6987	0.2291	80.30
OSR + RIQ	**46.57**	**1.50×**	39.10	5.74×	**27.25**	**0.7895**	**0.1105**	**81.68**
I2V								
Dense	70.22	1.00×	100.00	1.00×	∞	1.0000	0.0000	87.30
OSR + KIVI	56.32	1.25×	42.73	4.86×	25.25	0.7269	0.2286	82.74
OSR + QVG	50.75	1.38×	42.83	**6.14×**	24.93	0.7086	0.2473	80.09
OSR + RIQ	**47.93**	**1.47×**	42.66	5.74×	**28.13**	**0.8150**	**0.1056**	**87.11**

4.3 Component Evidence

We test the routing and representation choices motivated by the residual decomposition through three controlled comparisons. Appendix B.6 specifies the experiment configurations.

Correction-aware residual scoring. The score in Equation 10 augments the content term with an excess-mass proxy and a mass guard. With pooled correction fixed, full OSR achieves the lowest mean relative L2 error among four selectors across all five low-density targets, outperforming fixed-block SVG-EAR-style routing (Zhou et al., 2026) by 6.4–25.0% (Figure 5(a)). Adding the excess-mass proxy and mass guard to content-only scoring reduces error from 21.76% to 15.60% at 7.5% density, supporting their inclusion under tight attention budgets.

Source-aware residual budgeting. We next test whether global residual scores suffice for joint history/current routing. At matched retained-block densities and with the residual formulation and

correction fixed, source-aware OSR reduces mean relative output L2 error by 18.1–30.4% over global-only budgeting across six density points from 6% to 30% (Figure 5(b)). This supports accounting for source-relative importance when allocating token-level computation.

Preserving statistics and residual interactions. To test RIQ's mean–residual design, we start from K-MSE fitting and direct V quantization, then add block-residual V encoding, block-statistic restoration, and residual-interaction fitting (Figure 5(c)). Each step lowers mean E_D and E_Δ. With the V representation fixed, interaction fitting further reduces error relative to K-MSE fitting, supporting the use of K–V interactions beyond elementwise key reconstruction.

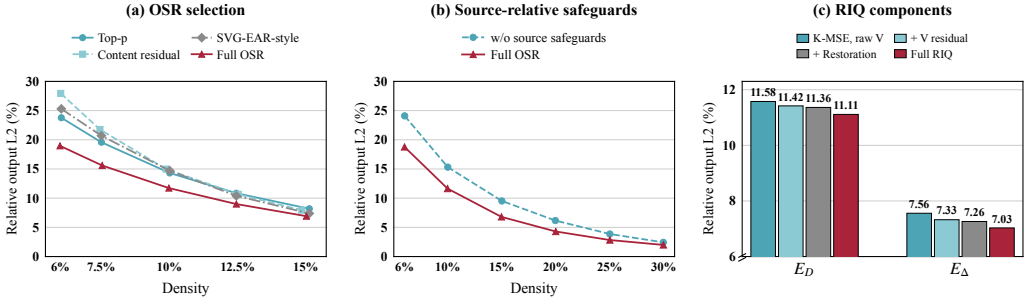


Figure 5: **Component evidence.** (a) Full OSR lowers mean output L2 error at actual retained block densities of 6–15%. (b) OSR with and without source-relative safeguards at matched aggregate 6–30% density. (c) RIQ representation ablation on 480p/10s T2V ($\lambda = 1$): E_Δ and E_D are relative L2 against BF16 OSR and dense outputs.

4.4 Kernel Efficiency

At scale 29 of 720p/5s, the complete OSR path—block statistics, residual scoring, route packing, and corrected sparse attention—achieves $9.75\times$, $5.23\times$, and $6.19\times$ speedups over FlashAttention-2 (Dao, 2024), FlashAttention-3 (Shah et al., 2024), and cuDNN attention (NVIDIA, 2026), respectively (Figure 6(a), scales 28–29). At scales 40–41 of 480p/10s, fusing RIQ decoding with KV layout preparation saves 2.89–4.03 ms per call over separate decoding and reduces operator latency by 12.8–14.0% relative to BF16 OSR (Figure 6(b)).

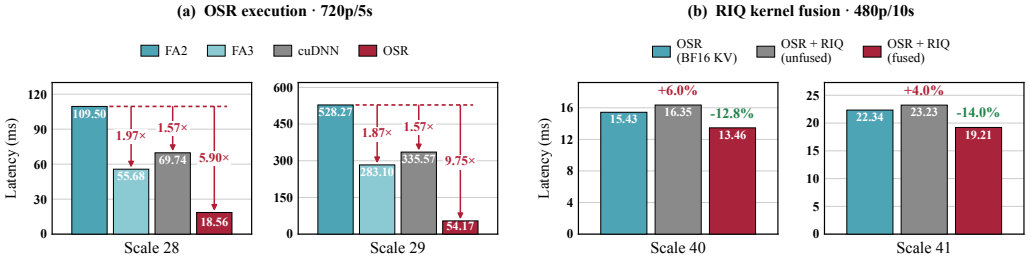


Figure 6: **Kernel performance across scales on H200.** (a) 720p/5s: latency; arrows show speedup over FA2 ($t_{\mathrm{FA2}}/t_{\mathrm{method}}$). (b) 480p/10s: latency of BF16 OSR and RIQ with separate or fused decoding/layout preparation; percentages show latency changes relative to BF16 OSR.

5 Conclusion and Future Work

We presented residual-centric inference for spacetime autoregressive video generation, using post-correction residuals to guide attention routing (OSR) and history quantization (RIQ). On InfinityStar, OSR achieves up to $1.86\times$ speedup ($2.01\times$ with FFN FP8), while OSR+RIQ achieves up to $1.50\times$ speedup and $5.74\times$ retained-history compression, with generation quality largely preserved. Our evaluation focuses on InfinityStar, reflecting the limited availability of spacetime autoregressive video backbones. However, the residual-centric formulation is not tied to InfinityStar's specific token layout or scale schedule: OSR relies on blockwise corrected attention under shared softmax normalization, while RIQ additionally assumes reusable historical KV. These conditions suggest broader applicability to architectures with attention bottlenecks and reusable context; future work will investigate this potential across different attention patterns and cache-reuse mechanisms.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript writing and language polishing, and with layout and visual refinement of selected figures. All technical content, numerical results, and scientific claims were produced and verified by the authors.

REPRODUCIBILITY STATEMENT

Section 3 presents the methods, with detailed pseudocode in Appendix A. Section 4.1 specifies the experimental setup, while Appendix B documents baseline adaptations, system optimizations, efficiency measurement procedures, and component evaluation protocols. Appendix C.7 details benchmark score aggregation and paired bootstrap uncertainty estimation.

REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240, 2024. doi: 10.52202/079017-3180.

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

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979. doi: 10.1214/aos/1176344552.

Hang Guo, Yawei Li, Taolin Zhang, Jiangshan Wang, Tao Dai, Shu-Tao Xia, and Luca Benini. FastVAR: Linear visual autoregressive modeling via cached token pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19011–19021, 2025.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise AutoRegressive modeling for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15733–15744, 2025.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303, 2024. doi: 10.52202/079017-0040.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(3):3268–3285, 2026. doi: 10.1109/TPAMI.2025.3633890.

Vinay Joshi, Pratik Prabhanjan Brahma, Zicheng Liu, and Emad Barsoum. TaDA: Training-free recipe for decoding with adaptive KV cache compression and mean-centering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 1435–1443, 2025. doi: 10.18653/v1/2025.acl-industry.101.

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.

Haopeng Li, Yitong Li, Junsong Chen, Tian Ye, Haozhe Liu, Jincheng Yu, Duomin Wang, Ruihua Zhang, Zeke Xie, Enze Xie, and Song Han. Sol-Attn: Accelerating video generation inference via on-the-fly attention sparsification. *arXiv preprint arXiv:2607.24027*, 2026a.

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*, 2026b.

Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. Memory-efficient visual autoregressive modeling with scale-aware KV cache compression. In *Advances in Neural Information Processing Systems*, volume 38, pp. 13736–13761, 2025a. doi: 10.52202/085713-0462.

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 *Advances in Neural Information Processing Systems*, volume 38, pp. 16822–16852, 2025b. doi: 10.52202/085713-0568.

Jinlai Liu, Jian Han, Bin Yan, Hui Wu, Fengda Zhu, Xing Wang, Yi Jiang, Bingyue Peng, and Zehuan Yuan. InfinityStar: Unified spacetime AutoRegressive modeling for visual generation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 170054–170072, 2025. doi: 10.52202/085713-5660.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *International Conference on Machine Learning*, pp. 32332–32344, 2024.

Rongge Mao, Chengqi Dong, and S Kevin Zhou. LazyVAR: Accelerating visual autoregressive models via scale-wise token pruning and parallel group decoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12129–12139, 2026.

NVIDIA. cuDNN documentation: Scaled dot product attention. `https://docs.nvidia.com/deeplearning/cudnn/latest/operations/Attention.html`, 2026. Accessed September 24, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, volume 37, pp. 68658–68685, 2024. doi: 10.52202/079017-2193.

Zunhai Su, Hanyu Wei, Zhe Chen, Wang Shen, Linge Li, Huangqi Yu, and Kehong Yuan. RotateKV: Accurate and robust 2-bit KV cache quantization for LLMs via outlier-aware adaptive rotations. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 6200–6208, 2025. doi: 10.24963/ijcai.2025/690.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Pardis Taghavi, Reza Langari, and Gaurav Pandey. Partition the support, reconstruct the residual: Training-free sparse attention for video generation and world models. *arXiv preprint arXiv:2608.18484*, 2026.

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.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems*, volume 37, pp. 84839–84865, 2024. doi: 10.52202/079017-2694.

Tuna Tuncer, Felix Becker, and Thomas Pfeil. Quantized keys steal attention: Bias correction for KV-cache compression in video diffusion. *arXiv preprint arXiv:2605.26266*, 2026.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero 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, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse VideoGen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *International Conference on Machine Learning*, pp. 68208–68224, 2025.

Haocheng Xi, Shuo Yang, Yilong Zhao, Muyang Li, Han Cai, Xingyang Li, Yujun Lin, Zhuoyang Zhang, Jintao Zhang, Xiuyu Li, Zhiying Xu, Jun Wu, Chenfeng Xu, Ion Stoica, Song Han, and Kurt Keutzer. Quant VideoGen: Auto-regressive long video generation via 2-bit KV-cache quantization. In *International Conference on Machine Learning*, 2026.

Rui Xie, Tianchen Zhao, Zhihang Yuan, Rui Wan, Wenxi Gao, Zhenhua Zhu, Xuefei Ning, and Yu Wang. LiteVAR: Compressing visual autoregressive modelling with efficient attention and quantization. *arXiv preprint arXiv:2411.17178*, 2024.

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 *Advances in Neural Information Processing Systems*, volume 38, pp. 96965–96991, 2025. doi: 10.52202/085713-3243.

Sungwoong Yune, Suheon Jeong, and Joo-Young Kim. FastSTAR: Spatiotemporal token pruning for efficient autoregressive video synthesis. *arXiv preprint arXiv:2603.07192*, 2026.

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 *International Conference on Machine Learning*, pp. 76397–76413, 2025a.

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 *Advances in Neural Information Processing Systems*, volume 38, pp. 152509–152534, 2025b. doi: 10.52202/085713-5099.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *International Conference on Machine Learning*, pp. 74714–74731, 2025c.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Xuanyi Zhou, Qiuyang Mang, Shuo Yang, Haocheng Xi, Jintao Zhang, Huanzhi Mao, Joseph E. Gonzalez, Kurt Keutzer, Ion Stoica, and Alvin Cheung. SVG-EAR: Parameter-free linear compensation for sparse video generation via error-aware routing. In *European Conference on Computer Vision*, pp. 167–185, 2026. doi: 10.1007/978-3-032-37359-5_10.

A Algorithm Details

A.1 OSR

Algorithm 1 summarizes routing and corrected sparse attention for one attention call. It combines block summaries with query statistics to score history and current blocks, then selects low-cost blocks for pooling under a unit omission budget. Token-level contributions from BF16 or reconstructed history KV are combined with pooled contributions from the original block means under a shared softmax normalization.

Algorithm 1 OSR: residual-aware routing and corrected sparse attention

Require: Queries Q; text, history, and current KV with native visibility; block size $B = 64$; content-weight parameter τ; mass guard γ; dispersion coefficient κ; $\epsilon = 10^{-12}$.
Ensure: Corrected sparse attention output O using the available KV representation.

Partition eligible visual KV into $\mathcal{B}_H, \mathcal{B}_C$; let $\mathcal{E}$ contain non-sparsified text and protected tails.
Obtain original block means, counts, and (W_b, U_b, λ_b) using Equation 8; reuse cached history summaries.
for each query head and query block **do**
    Compute mean $\bar{q}$ and diagonal variance σ_q^2 of $Q/\sqrt{d}$, pooling CFG branches for their shared route.
    Compute r_b for each candidate block.
    for each nonempty domain $\mathcal{D} \in \{\text{all visible context}, \mathcal{B}_H, \mathcal{B}_C\}$ **do**
        Compute $O_0^{\mathcal{D}}, \bar{p}_b^{\mathcal{D}}$ by Equation 9.
        Compute $A_b^{\mathcal{D}}, B_b^{\mathcal{D}}$ using that domain's anchor and mass.
        $\widetilde{\rho}_b^{\mathcal{D}} \leftarrow A_b^{\mathcal{D}}/\tau + B_b^{\mathcal{D}} + \gamma \bar{p}_b^{\mathcal{D}}$.
    end for
    $\widetilde{c}_b \leftarrow \max(\widetilde{\rho}_b^{\text{global}}, \widetilde{\rho}_b^{s(b)})$ for $b \in \mathcal{B}_H \cup \mathcal{B}_C$.
    Bin $\widetilde{c}_b$ into 256 ordered logarithmic bins.
    Set normalized omission budget to 1.
    $\mathcal{U} \leftarrow$ blocks in the lowest complete bins satisfying $\sum_{b\in\mathcal{U}} \widetilde{c}_b \le 1$.
    Scan the next bin in native block order; add blocks while their cost fits $1 - \sum_{b\in\mathcal{U}} \widetilde{c}_b$.
    Remove the highest-cost block and other mandatory retained blocks from $\mathcal{U}$; $\mathcal{S} \leftarrow (\mathcal{B}_H \cup \mathcal{B}_C) \setminus \mathcal{U}$.
    Pack the shared route $\mathcal{S}$ for token-level attention.
    for each query Q_i and CFG branch **do**
        $q \leftarrow Q_i/\sqrt{d}$; $\mathcal{T} \leftarrow \mathcal{E} \cup \text{tokens}(\mathcal{S})$.
        Obtain $K_j^{\text{read}}, V_j^{\text{read}}$ for $j \in \mathcal{T}$: BF16 KV, or Algorithm 2's sparse view for compressed history.
        $Z \leftarrow \sum_{j\in\mathcal{T}} e^{q^\top K_j^{\text{read}}} + \sum_{b\in\mathcal{U}} n_b e^{q^\top \bar{K}_b}$.
        $N \leftarrow \sum_{j\in\mathcal{T}} e^{q^\top K_j^{\text{read}}} V_j^{\text{read}} + \sum_{b\in\mathcal{U}} n_b e^{q^\top \bar{K}_b} \bar{V}_b$.
        $O_i \leftarrow N/Z$.
    end for
end for
return O.

A.2 RIQ

Algorithm 2 separates one-time history encoding from cache reads. Encoding stores original block summaries and packed INT2 codes, fitting key scales against the original value residuals as in Equation 12. Reads reconstruct KV using the stored block means and value RMS. Dense and sparse reads share codes but use reconstruction-oriented and interaction-fitted key scales, respectively.

Algorithm 3 gives the per-group affine quantizer used for both K and V. It alternates code assignment with least-squares scale and offset fitting, then reassigns codes using the stored FP16 parameters.

Algorithm 2 RIQ: INT2 history encoding and reconstruction

Require: Historical KV to cache; $B = 64$; channel group size $G = 64$; regularization weight $\lambda = 1$; numerical stabilizer ϵ.
Ensure: Packed history with original summaries and dense/sparse read views.
AFFINEFIT is specified in Algorithm 3.

function WRITERIQ(K, V)
  for each complete B-token history block and KV head **do**
    Compute original means μ_K, μ_V and OSR moments; store means in BF16 and retain the original routing header.
    $R_K \leftarrow K - \mathbf{1}\,\mathrm{mean}_i(K_i)^\top$; $R_V \leftarrow V - \mathbf{1}\,\mathrm{mean}_i(V_i)^\top$ in FP32.
    $r_V \leftarrow \mathrm{FP16}(\sqrt{\max(\mathrm{mean}_{i,c}\, V_{ic}^2 - \mathrm{mean}_c\, \mu_{V,c}^2, 0)})$.
    $(C_K, a_K, z_K) \leftarrow$ AFFINEFIT($K, \{1/8, 3/8, 5/8, 7/8\}, 3$) per channel over B tokens.
    $T \leftarrow C_K - \mathbf{1}\,\mathrm{mean}_i(C_{K,i})^\top$; $s_{\mathrm{old}} \leftarrow a_K/4$.
    Fit and clip s by Equation 13 using T, R_K, original R_V, and λ.
    $a_{\mathrm{dense}} \leftarrow a_K$; $a_{\mathrm{sparse}} \leftarrow \mathrm{FP16}(4s)$.
    $X_V \leftarrow \mathrm{BF16}(V - \mathbf{1}\mu_V^\top)$.
    $(C_V, s_V, z_V) \leftarrow$ AFFINEFIT($X_V, \{0, 1, 2, 3\}, 3$) per token over G channels.
    Pack C_K, C_V with four 2-bit codes per byte: $P_j = \sum_{r=0}^{3} C_{4j+r}\, 2^{2r}$.
    Store $P_K, P_V, a_{\mathrm{dense}}, a_{\mathrm{sparse}}, s_V, z_V, r_V$ and the shared header; discard temporary z_K, T, R_K, R_V.
  end for
  Keep text, current KV, and incomplete/protected tails in BF16.
  return Packed history and headers.
end function
function READRIQ(packed block, view)
  Unpack C_K, C_V; $T \leftarrow C_K - \mathbf{1}\,\mathrm{mean}_i(C_{K,i})^\top$.
  $a \leftarrow a_{\mathrm{sparse}}$ for corrected sparse attention; otherwise $a \leftarrow a_{\mathrm{dense}}$.
  $\widehat{K} \leftarrow \mathbf{1}\mu_K^\top + T\,\mathrm{diag}(a/4)$.
  $\widetilde{R}_{V,ic} \leftarrow z_{V,i,g(c)} + s_{V,i,g(c)} C_{V,ic}$ for each token i and channel c in group $g(c)$.
  $R_V^{\mathrm{ctr}} \leftarrow \widetilde{R}_V - \mathbf{1}\,\mathrm{mean}_i(\widetilde{R}_{V,i})^\top$; $\widetilde{r}_V^2 \leftarrow \mathrm{mean}_{i,c}(R_{V,ic}^{\mathrm{ctr}})^2$.
  $\widehat{V} \leftarrow \mathbf{1}\mu_V^\top + r_V R_V^{\mathrm{ctr}} / \sqrt{\widetilde{r}_V^2 + \epsilon}$.
  return $\mathrm{BF16}(\widehat{K}), \mathrm{BF16}(\widehat{V})$; retain the original correction header.
end function

Algorithm 3 RIQ affine-fitting subroutine (per quantization group)

Require: Group X; levels $L = \{1/8, 3/8, 5/8, 7/8\}$ for K or $L = \{0, 1, 2, 3\}$ for V; iteration count $J = 3$.
Ensure: Integer indices C and stored FP16 affine parameters (a, z).

function AFFINEFIT(X, L, J)
  $X \leftarrow \mathrm{FP32}(X)$; $z \leftarrow \min X$; $a \leftarrow \max(\max X - \min X, 10^{-8})$.
  if $L = \{0, 1, 2, 3\}$ **then**
    $a \leftarrow a/3$.
  end if
  for $j = 1, \ldots, J$ **do**
    $C_i \leftarrow \arg\min_{k \in \{0,1,2,3\}} |X_i - (z + aL_k)|$; $y_i \leftarrow L_{C_i}$.
    $a' \leftarrow \mathrm{Cov}(X, y) / \max(\mathrm{Var}(y), 10^{-8})$.
    if $\mathrm{Var}(y) > 10^{-8}$ and $|a'| > 10^{-8}$ **then**
      $a \leftarrow a'$; $z \leftarrow \mathrm{mean}(X) - a\,\mathrm{mean}(y)$.
    end if
  end for
  $z \leftarrow \mathrm{FP16}(z)$; $a \leftarrow \max(\mathrm{FP16}(a), 2^{-24})$.
  Reassign $C_i \leftarrow \arg\min_{k \in \{0,1,2,3\}} |X_i - (z + aL_k)|$ using stored parameters.
  return (C, a, z).
end function

B Implementation and Baseline Details

This section supplements the configurations in Section 4.1: efficiency measurements, common System optimizations, sparse-attention adaptations, the FastSTAR reimplementation, and history-quantization baselines.

B.1 Efficiency Measurement Protocol

Global attention density. For each native attention call c, let P_c^{native} count query–key token pairs under InfinityStar's native visibility, and P_c^{token} count those retained for token-level computation. We report

$$D_{\text{global}} = 100 \times \frac{\sum_c P_c^{\text{token}}}{\sum_c P_c^{\text{native}}} \ \%.$$

Counts include batch/CFG branches and query heads, valid token lengths, all executed scales, layers, and native repetitions. The sums aggregate calls in the density-diagnostic runs, rather than averaging per-call percentages. Unmodified calls contribute full native work; dense text, history where unsparsified, and protected tails remain in both counts. For token pruning, the numerator uses the reduced executed query/key lengths, while the denominator uses the unpruned native lengths. Pooled-correction proxy interactions and routing/statistics computation are excluded from this token-pair ratio, but their execution time is included in latency. Global density therefore differs from eligible-block retention in the component studies and is not a total-FLOP ratio.

End-to-end latency. A synchronized wall-clock timer surrounds the generation call, starting before prompt encoding (and I2V conditioning preparation) and ending after decoded frames are returned. It includes text tokenization and encoding, I2V image transformation and VAE conditioning encoding, autoregressive generation, routing, token-level attention and pooled correction, KV encoding/decoding, and VAE decoding/postprocessing. CPU–GPU transfers performed within this call, including runtime model paging and conditioning transfers, are included. Model loading, one-time setup, and any VAE offloading performed before the timer are excluded, as are input-file loading, final frame transfer to the host, video serialization, and quality evaluation.

Timing procedure. Each generation worker performs one full-shape warm-up before measured requests and reuses the loaded model. Warm-up time is excluded, and prompt-dependent sparse state is reset where required. We synchronize the GPU immediately before starting and after completing each timed call. Each evaluated input/configuration receives one measured generation; reported latency is the arithmetic mean of these request times, not the minimum or median of repeated trials on the same input. Speedup is the corresponding Dense mean divided by the method mean. Route-density logging and per-attention timing instrumentation are disabled in latency runs. Density and operator-level profiles are collected separately, so their diagnostic overhead is not charged to end-to-end timing.

B.2 Common System Optimizations

All accelerated comparison methods use the same common System optimizations. GQA-aware attention consumes the native KV-head layout without materializing copies for each query head. To limit activation memory, FFNs run in 8,192-token chunks and standalone RMS normalization in 16,384-token chunks. The post-attention sequence—residual addition, RMS normalization, FFN, and the second residual addition—is also executed within one 8,192-token chunk at a time, avoiding full-sequence intermediate tensors between these operations.

KV entries are released once the native scale schedule no longer requires them; history that remains visible to later attention is retained. The VAE decoder is offloaded from the GPU during autoregressive generation and brought back for tiled BF16 decoding. Together, these changes reduce unnecessary copies, temporary activations, and overlapping memory residency, with minimal impact on generation quality and end-to-end latency. Appendix C.1 reports their combined effect.

Optional FFN FP8. Only rows marked OSR + FP8 enable this additional optimization. At 720p/5s scales 27–29, all three SwiGLU projections—gate, up, and down—use NVIDIA Trans-

former Engine's E4M3 block-scaled FP8 GEMMs. The recipe uses one-dimensional activation blocks, two-dimensional weight blocks, and FP32 scaling factors. Activations are quantized at runtime; the projections return BF16 tensors for the activation, gating, and residual operations. Earlier scales use BF16 FFNs, and their original weights are retained across generations. Attention and KV remain BF16 in these 720p/5s runs. The FP8 option uses the same OSR routing policy and is separate from the common System optimizations above.

B.3 Adapting Diffusion Sparse Attention to InfinityStar

Shared adaptation. At 720p/5s, PISA, Sol-Attn, and SVG2 sparsify current visual KV at scales 27–29; earlier scales, history, and text remain dense. We preserve each method's selector and correction, adapt the native GQA layout, and merge sparse current attention with dense history/text using log-sum-exp normalization. All three use BF16 KV and FFNs. Keeping history dense preserves their original current-grid selection and execution optimizations without adding a cross-source routing policy. History still participates through the dense attention branch; it is not ignored.

PISA. We retain PISA's HYD selector and correction (Li et al., 2026b), with default density parameter $d = 0.017$.

Sol-Attn. We use the upstream selector, online routing, and zeroth-order correction (Li et al., 2026a), with $\tau_{\text{Sol}} = 1.7$, diagonal thresholding, and one KV split.

SVG2. We retain per-head clustering, semantic permutation, and the FlashInfer executor (Yang et al., 2025), using 300/1,000 query/key clusters, top-$p = 0.20$, and a 0.10 minimum cluster-retention ratio. Layer 0 remains dense. K-means uses 50 initial and two reuse iterations; centroids are cached separately for each layer and scale. Timing includes clustering and layout transformations.

B.4 FastSTAR Reimplementation and Configuration

We reimplement FastSTAR from its paper (Yune et al., 2026), as official code was unavailable at the time of our experiments. The main-table rows use the pruning schedule $(0.2, 0.3, 0.4, 0.55)$. The diagnostic mean token retention is 64.70% (35.30% token pruning). The main table reports global QK retention, including protected regions and sources, rather than the selector's token-pruning fraction.

B.5 Quantization Baselines on the Same OSR Policy

The 480p/10s comparison fixes OSR at scales 27 and 39–41 and compresses only the large history cache written at scale 27. All codecs retain the original shared OSR statistics.

KIVI (Liu et al., 2024) uses INT2 with 32-element groups and a residual length of 128, retaining its native K prefill grouping, the latest 128 V tokens in BF16, and the exact tails required by OSR. QVG (Xi et al., 2026) uses native single-stage INT2 product/residual quantization with 256 centroids, two fitting iterations, 64-token blocks, and FP8 scales; its clustering random state is isolated from generation. Neither baseline receives RIQ's mean/RMS restoration.

Both codecs use the common OSR executor and reconstruct the full history before attention. RIQ instead uses the fused decoding and layout preparation described in Section 4.4. The comparison therefore measures codec fidelity and deployed system efficiency, rather than equal optimization effort for each decoding kernel.

B.6 Component Evaluation Details

OSR selection and source-aware budgeting. Figures 5(a,b) use six diagnostic 720p/5s T2V inputs, scales 27–29, and zero-based layers 5/11/17/23/29/35 (144 calls, including native repetitions). Each call evaluates 16 stratified complete 64-token query blocks, all heads and both CFG branches, against full BF16 KV. All variants share pooled correction, protected text/tails, and output normalization. Panel (a) compares Top-p, content-residual scoring, full OSR, and a fixed-block SVG-EAR-style control without semantic clustering or permutation; it does not reproduce the complete SVG-

EAR system. Panel (b) holds the full OSR score fixed and removes only source-relative safeguards. We scan each method's native p or τ using aggregate retained-block density alone and freeze parameters before evaluating errors. Density excludes protected text and tails, unlike main-table global QK density. Relative output L2 is $\|O - O_D\|_F / \|O_D\|_F$, aggregated equally over repetitions within layer–scale cells, then layers, scales, and inputs.

RIQ representation controls. Figure 5(c) uses 12 confirmation T2V inputs, disjoint from the sensitivity inputs used to choose λ in Table 7. After that scan selects $\lambda = 1$, we freeze it for this representation ablation. Each input contributes 30 native 480p/10s calls at layers $\{0, 9, 18, 27, 35\}$ and scales $\{39, 40, 41\}$, with eight evenly spaced complete query blocks per call, full KV, all heads, and both CFG branches. Diagnostics share a BF16-cache OSR trajectory and do not feed their outputs back into generation. The four controls successively use direct V quantization, residual V encoding, K/V mean and V-residual RMS restoration, and interaction-fitted K scales. The affine family, three fitting iterations, original routing and correction summaries, and OSR masks are fixed; V codes change only when switching from direct to residual encoding. The first three controls share K-MSE scales, and the last two share restored V. Restoration changes decoded tokens, not the original statistics used for the fixed routes. We report

$$E_\Delta = \frac{\|O_Q - O_B\|_F}{\|O_B\|_F}, \qquad E_D = \frac{\|O_Q - O_D\|_F}{\|O_D\|_F},$$

where O_B, O_Q, and O_D are BF16 OSR, quantized OSR, and dense outputs on the same Q/K/V. Errors use the same hierarchical averaging as the sensitivity scan.

C Supplementary Experiments

These experiments extend the main results and component controls in Sections 4.2–4.4. We isolate System and memory effects, examine current-only routing and RIQ regularization, compare attention–quantization configurations, and report speed–quality trade-offs and benchmark uncertainty.

C.1 System Ablation and Memory Breakdown

We isolate the common optimizations described in Appendix B.2. Table 3 evaluates the common optimizations in Appendix B.2 with dense attention and BF16 KV/FFNs, using the same inputs and generation settings as Dense.

Table 3: **Effect of common System optimizations on H200.** Time and peak-memory deltas are System minus Dense; parentheses show memory changes relative to Dense. PSNR, SSIM, and LPIPS measure fidelity to Dense videos. ΔVB is a percentage of the 100-point maximum.

Workload	ΔTime (s)↓	ΔPeak (GiB / %)↓	PSNR (dB)↑	SSIM↑	LPIPS↓	ΔVB↑
720p/5s T2V	−0.07	−28.64 (−49.1%)	40.00	0.9655	0.0184	+0.059%
720p/5s I2V	−1.00	−28.59 (−49.0%)	39.23	0.9671	0.0151	−0.025%
480p/10s T2V	−1.11	−12.38 (−27.4%)	38.10	0.9646	0.0131	+0.002%
480p/10s I2V	−0.78	−12.38 (−27.4%)	39.02	0.9685	0.0131	−0.018%

History storage and inference memory. Table 4 separates retained history from whole-inference memory. System reduces total memory without compressing history; OSR keeps BF16 history with routing statistics, while RIQ compresses this persistent representation. The $5.74\times$ KV compression ratio uses OSR's BF16-history reference, including its shared statistics (8.9071/1.5507 GiB), rather than Native Dense's 8.73 GiB of KV tensors alone. Together, these optimizations reduce peak inference allocated memory from 45.20 to 26.60 GiB (41.1%). This reduction reflects both common System optimizations and persistent-history compression.

Table 4: **Memory breakdown at 480p/10s T2V on H200 (GiB).** Retained history includes Clip 1 visual KV and persistent metadata, excluding text/current KV; BF16-only storage is calculated from tensor shapes. Overall peak is inference allocated memory, excluding loading and warm-up. RIQ uses $\lambda = 1$.

Configuration	Retained history	Overall peak allocated memory
Native Dense	8.73	45.20
System + dense attention	8.73	32.82
OSR + BF16 history	8.91	33.96
OSR + RIQ	1.55	26.60

C.2 Current-only Attention Control

To control for sparse-source scope in Table 1, we evaluate OSR on 720p/5s T2V with only current visual KV sparsified, keeping history dense as in our PISA, Sol-Attn, and SVG2 adaptations. This configuration achieves $1.85\times$ speedup, compared with $1.86\times$ when history is also sparsified (30.57 versus 30.35 s; Table 5). History sparsification therefore contributes little additional acceleration in this short-video setting.

Table 5: **Current-only sparse attention at 720p/5s T2V.** The first four methods sparsify current visual KV at scales 27–29 while keeping history and text dense. All share System optimizations and BF16 KV/FFNs. OSR (current-only) uses $\tau = 1.6$, $\gamma = 0.3$, and 64-token blocks. The last row is the default history-and-current OSR for reference. Density is global attention density; configurations retain their native parameters rather than matching density. Speedup and fidelity use Native Dense as the reference. Main-table rows are repeated for comparison.

Method	Time (s)↓	Speedup↑	Density (%)	PSNR↑	SSIM↑	LPIPS↓	VBench↑
PISA	33.64	1.68×	23.30	30.36	0.8817	0.0696	84.09
Sol-Attn	33.34	1.69×	23.95	29.80	0.8714	0.0771	83.99
SVG2	81.80	0.69×	27.73	30.88	0.8841	0.0678	84.03
OSR (current-only)	30.57	1.85×	23.20	31.26	0.8921	0.0625	84.71
OSR	30.35	1.86×	19.45	32.31	0.9056	0.0537	84.82

C.3 OSR Parameter Sensitivity

Protocol. We vary the mass-guard weight $\gamma \in \{0, 0.15, 0.3, 0.6\}$ at $\kappa = 0.5$, and the pooled-mass dispersion coefficient $\kappa \in \{0, 0.25, 0.5, 1\}$ at $\gamma = 0.3$. All seven settings use the same scoring formulation, source-aware budgeting, pooled correction, and protected text/tails. We reuse six diagnostic 720p/5s T2V inputs and 96 dense-trajectory calls: scales 27–29, zero-based layers $\{0, 1, 18, 35\}$, including native repetitions, with six evenly spaced complete 64-token query blocks per call, all heads, both CFG branches, and full KV. All configurations use identical Q/K/V. Relative output L2 is averaged over repetitions within each layer–scale cell, then layers, scales, and inputs with equal weights at each level.

Parameter sensitivity. With $\tau = 1.6$ fixed, increasing either γ or κ makes routing more conservative, retaining more blocks and reducing attention-output error (Table 6). These parameters therefore control the trade-off between token-level computation and approximation fidelity. We use $(\gamma, \kappa) = (0.3, 0.5)$ as a moderate operating point throughout the main experiments, yielding 12.65% eligible-block density and 12.116% relative output error on this diagnostic set.

C.4 RIQ Regularization Sensitivity

Protocol. We use 12 sensitivity inputs per task, excluding the earlier RIQ development inputs. The scan evaluates $\lambda \in \{0.1, 0.3, 1, 3, 10, 30\}$ against K-MSE fitting. All variants share integer K codes, block-residual V, original block statistics, and fixed OSR routes; only K scales change. We measure 30 native attention calls per input at layers $\{0, 9, 18, 27, 35\}$ and scales $\{39, 40, 41\}$, sampling eight

Table 6: **Sensitivity to OSR routing parameters.** One-factor sweeps around $(\gamma, \kappa) = (0.3, 0.5)$ with $\tau = 1.6$ fixed. Density measures eligible-block retention, excluding protected text/tails and full-video dense calls. L2 denotes relative attention-output error. Results use six diagnostic T2V inputs and 96 attention calls; Appendix C.3 specifies sampling. Bold parameters indicate the deployed default.

γ	κ	Density (%)	L2 (%)↓
0	0.5	10.72	15.303
0.15	0.5	11.79	13.305
0.3	**0.5**	12.65	12.116
0.6	0.5	14.12	10.625
0.3	0	1.93	27.607
0.3	0.25	7.56	16.374
0.3	1	21.75	7.368

complete query blocks per call. Errors are averaged over repetitions within each layer–scale cell, then equally over layers, scales, and inputs.

K-MSE fitting control. K-MSE fits each key channel's scale to minimize $\|T\,\mathrm{diag}(s) - R_K\|_F^2$ with the INT2 codes fixed. Its solution is $s_c^{\mathrm{MSE}} = t_c^\top r_{K,c} / \max(t_c^\top t_c, \epsilon)$, clipped to the same $[0.5 s_{\mathrm{old},c}, 1.5 s_{\mathrm{old},c}]$ interval as RIQ. This is the reconstruction-only counterpart of Equation 12: it removes the interaction term while retaining the same V representation, original summaries, and routes.

Sensitivity. Table 7 shows that $\lambda = 1$ minimizes the equally weighted T2V/I2V mean E_Δ (6.9539%) among tested values. We select this value on the sensitivity inputs and keep it fixed for the progressive comparison in Figure 5(c), which uses 12 disjoint T2V confirmation inputs. The table reports parameter selection, whereas the figure evaluates the representation components after selection; their absolute errors are not directly interchangeable.

Table 7: **RIQ regularization sensitivity.** Relative output L2 errors (%) on 12 sensitivity inputs per task. Avg. is the equally weighted T2V/I2V mean of E_Δ, used to select $\lambda = 1$. K-MSE is fixed-code, reconstruction-only K-scale fitting with the same clipping rule (Appendix C.4). Figure 5(c) instead uses disjoint confirmation inputs with $\lambda = 1$ fixed; the error definitions are shared.

	E_Δ (%)↓			E_D (%)↓	
K-scale fit	T2V	I2V	Avg.	T2V	I2V
K-MSE fit	7.2818	7.0453	7.1635	11.3991	11.3890
$\lambda = 0.1$	7.0938	7.0061	7.0499	11.1290	11.2206
$\lambda = 0.3$	7.0450	6.9555	7.0002	11.1162	11.2029
$\lambda = 1$	7.0308	6.8770	**6.9539**	11.1329	11.1908
$\lambda = 3$	7.0827	6.8762	6.9795	11.1983	11.2257
$\lambda = 10$	7.1777	6.9439	7.0608	11.2950	11.2979
$\lambda = 30$	7.2348	7.0015	7.1182	11.3548	11.3489

C.5 Attention–Quantization Configuration Ablation

Table 8 crosses System (dense attention) and OSR with four history-cache representations. Every configuration uses the same system optimizations and BF16 FFNs. Quantization is applied to the history written at scale 27; OSR sparsifies scales 27 and 39–41. RIQ uses $\lambda = 1$. For the System + RIQ control only, we retain OSR + RIQ's read schedule despite using dense attention: interaction-fitted K scales at 39–41 and reconstruction scales elsewhere. This controlled exception keeps the reconstructed history representation unchanged when isolating the effect of sparse attention. All fidelity metrics retain the original Dense videos as their common reference, so System with dense attention need not have zero error.

With BF16 history, OSR reduces latency from 68.78 to 49.08 seconds in T2V and from 69.45 to 49.59 seconds in I2V. At fixed attention mode, RIQ retains higher PSNR and lower LPIPS than KIVI and QVG while compressing history by $5.74\times$. This advantage holds under both dense and corrected

sparse attention. These controls establish compatibility; they do not by themselves demonstrate a synergistic interaction. Timings include cache encoding and decoding and come from independent end-to-end runs, rather than isolated codec-overhead measurements.

Table 8: **Attention–quantization configurations at 480p/10s.** All rows use the same system optimizations and BF16 FFNs; System denotes dense attention with these optimizations, distinct from Native Dense in the main table. PSNR and LPIPS use the original unquantized Dense videos as their common reference. KV compression measures retained-history storage relative to BF16, including codec metadata. Existing rows are repeated for complete comparison; RIQ uses $\lambda = 1$ throughout.

T2V Configuration	History KV	Time (s)↓	KV comp.↑	PSNR↑	LPIPS↓	VBench↑
System	BF16	68.78	1.00×	38.10	0.0131	81.89
OSR	BF16	49.08	1.00×	27.58	0.0999	82.30
System	KIVI INT2	74.43	4.86×	26.99	0.2022	80.84
OSR	KIVI INT2	56.50	4.86×	25.08	0.2096	80.97
System	QVG INT2	67.96	6.14×	26.73	0.2245	79.25
OSR	QVG INT2	52.26	6.14×	24.79	0.2291	80.30
System	RIQ INT2	66.24	5.74×	29.41	0.0979	82.38
OSR	RIQ INT2	46.57	5.74×	27.25	0.1105	81.68
I2V Configuration	History KV	Time (s)↓	KV comp.↑	PSNR↑	LPIPS↓	VBench-I2V↑
System	BF16	69.45	1.00×	39.02	0.0131	87.28
OSR	BF16	49.59	1.00×	28.31	0.0980	87.46
System	KIVI INT2	75.10	4.86×	27.00	0.2291	81.90
OSR	KIVI INT2	56.32	4.86×	25.25	0.2286	82.74
System	QVG INT2	69.25	6.14×	26.99	0.2356	79.87
OSR	QVG INT2	50.75	6.14×	24.93	0.2473	80.09
System	RIQ INT2	67.72	5.74×	30.31	0.0926	87.21
OSR	RIQ INT2	47.93	5.74×	28.13	0.1056	87.11

C.6 System-Level Speed–Quality Trade-off

End-to-end deployment comparison. We compare two complete inference configurations on 720p/5s T2V: our FastSTAR reimplementation, and our OSR-based system with FFN FP8. Both use the common System optimizations in Appendix B.2, the same inputs and generation settings, and the same dense timing reference. The comparison characterizes the end-to-end trade-offs of these configurations.

Figure 7 shows three evaluated operating points per configuration, spanning measured speedups of $1.68\times$–$2.01\times$. FastSTAR varies its final pruning ratio while keeping the preceding three ratios at $(0.2, 0.3, 0.4)$. OSR varies τ, retaining $\gamma = 0.3$, 64-token blocks, historical and current sparsification. Both panels use measured end-to-end speedup on the horizontal axis; the vertical axes report VBench total and frame-averaged PSNR relative to Dense. Under the evaluated deployment configurations, our OSR-based system achieves slightly higher end-to-end speedup than our FastSTAR reimplementation, while maintaining higher VBench scores and better fidelity to dense outputs.

C.7 Benchmark Decomposition and Uncertainty

Score decomposition. Tables 9 and 10 decompose the main-table results, the System controls, and the additional dense-attention quantization configurations using official normalization and weights. On the 0–100 scale, T2V uses $\mathrm{Total} = (4\,\mathrm{Quality} + \mathrm{Semantic})/5$, whereas I2V uses $\mathrm{Total} = (\mathrm{Quality} + \mathrm{Alignment})/2$. T2V aggregates seven quality and nine semantic dimensions; I2V aggregates six quality dimensions and three alignment dimensions (image–subject, image–background, and camera motion), with camera-motion weight 0.1.

Paired bootstrap. We compute pointwise 95% percentile intervals (Efron, 1979) from 10,000 bootstrap replicates of the evaluated inputs for each workload. The sampling unit is a prompt for T2V and an image–prompt pair for I2V. Inputs are stratified by their evaluated dimension membership, preserving each dimension's sample count. Within a replicate, all methods share the sampled

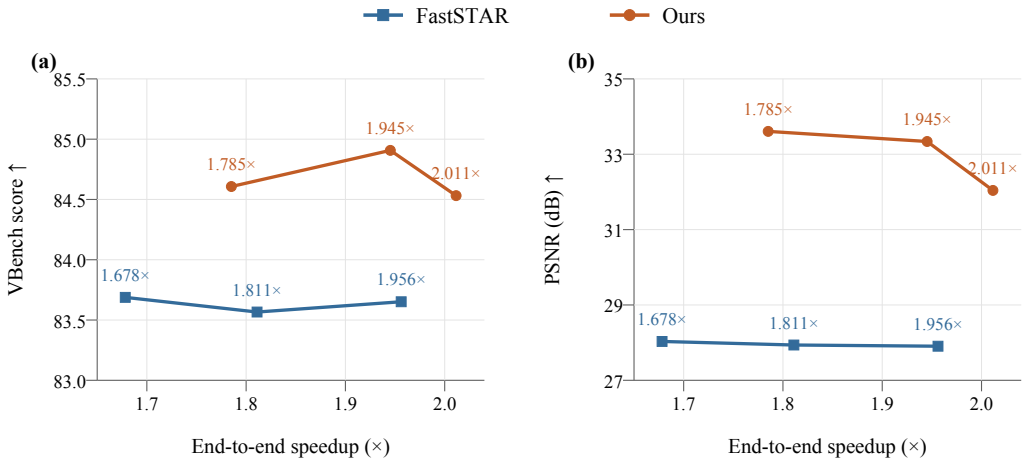


Figure 7: **Speed–quality trade-off on 720p/5s T2V.** The same three selected operating points per method are evaluated by (a) VBench total and (b) frame-averaged PSNR relative to Dense. Lines connect measurements; all points use the same inputs and dense reference.

input multiplicities, and an input shared by several dimensions receives the same multiplicity in each. We recompute the official dimension normalization, group scores, and total for every replicate; ΔTotal is paired against Dense. The intervals quantify variability across the evaluated inputs.

Interpretation. At 720p/5s, OSR differs from Dense by $+0.056/-0.035$ total-score points for T2V/I2V; with FFN FP8, the differences are $-0.230/-0.271$. At 480p/10s, OSR with RIQ differs by $-0.216/-0.191$ points. The intervals for I2V OSR with FP8 and T2V OSR with RIQ are below zero, indicating small measured score reductions. An interval containing zero does not establish equivalence; the decompositions and paired differences contextualize the aggregate scores alongside the fidelity metrics in the main tables.

Table 9: **720p/5s benchmark decomposition.** Entries show scores and pointwise 95% bootstrap intervals. All scores use the 0–100 scale; ΔTotal is the paired difference from Dense in score points. System retains dense attention and BF16 KV/FFNs.

T2V Method	Quality↑	Semantic↑	Total↑	ΔTotal
Dense	83.96 [81.51, 86.37]	87.96 [83.97, 91.35]	84.76 [82.59, 86.91]	–
System	83.99 [81.55, 86.40]	88.15 [84.12, 91.66]	84.82 [82.65, 86.95]	+0.059 [-0.128, +0.316]
PISA	83.13 [80.61, 85.64]	87.91 [83.91, 91.29]	84.09 [81.87, 86.30]	-0.675 [-1.983, +0.091]
Sol-Attn	83.13 [80.62, 85.60]	87.43 [82.73, 91.59]	83.99 [81.72, 86.25]	-0.775 [-2.197, +0.149]
SVG2	83.05 [80.52, 85.56]	87.94 [83.81, 91.39]	84.03 [81.83, 86.23]	-0.732 [-2.056, +0.043]
FastSTAR	82.72 [80.29, 85.15]	87.58 [83.23, 91.22]	83.69 [81.50, 85.85]	-1.074 [-2.498, -0.132]
OSR	83.82 [81.35, 86.24]	88.82 [84.78, 92.22]	84.82 [82.67, 86.96]	+0.056 [-0.206, +0.377]
OSR + FP8	83.83 [81.35, 86.27]	87.33 [83.04, 91.16]	84.53 [82.32, 86.71]	-0.230 [-0.567, +0.133]
I2V Method	Quality↑	Alignment↑	Total↑	ΔTotal
Dense	80.89 [79.75, 81.98]	96.01 [94.74, 97.26]	88.45 [87.60, 89.31]	–
System	80.85 [79.71, 81.93]	96.00 [94.73, 97.25]	88.42 [87.57, 89.28]	-0.025 [-0.055, +0.004]
PISA	79.11 [77.87, 80.33]	95.07 [93.77, 96.38]	87.09 [86.18, 88.01]	-1.355 [-1.877, -0.929]
Sol-Attn	80.58 [79.44, 81.69]	95.53 [94.24, 96.80]	88.05 [87.19, 88.93]	-0.394 [-0.811, -0.126]
SVG2	80.56 [79.42, 81.67]	95.51 [94.23, 96.80]	88.04 [87.17, 88.91]	-0.412 [-0.824, -0.138]
FastSTAR	79.72 [78.54, 80.88]	95.25 [93.95, 96.54]	87.48 [86.60, 88.36]	-0.965 [-1.421, -0.602]
OSR	80.85 [79.70, 81.95]	95.98 [94.71, 97.23]	88.41 [87.56, 89.27]	-0.035 [-0.073, +0.004]
OSR + FP8	80.71 [79.57, 81.82]	95.64 [94.36, 96.92]	88.18 [87.32, 89.05]	-0.271 [-0.687, -0.009]

Table 10: **480p/10s benchmark decomposition.** Notation and bootstrap protocol follow Table 9. System + codec rows retain dense attention with System optimizations. RIQ uses $\lambda = 1$, consistently with the main table and Table 8.

T2V Method	Quality↑	Semantic↑	Total↑	ΔTotal
Dense	81.24 [78.49, 83.84]	84.51 [79.08, 89.65]	81.89 [79.42, 84.27]	–
System	81.29 [78.54, 83.88]	84.32 [78.69, 89.51]	81.89 [79.42, 84.28]	+0.002 [-0.137, +0.132]
OSR	81.94 [79.19, 84.44]	83.71 [78.16, 88.96]	82.30 [79.73, 84.68]	+0.406 [-0.358, +1.723]
OSR + KIVI	80.65 [77.78, 83.19]	82.24 [76.72, 87.58]	80.97 [78.32, 83.33]	-0.925 [-2.368, +0.858]
OSR + QVG	79.73 [76.99, 82.35]	82.60 [76.98, 87.83]	80.30 [77.76, 82.70]	-1.591 [-2.607, -0.102]
System + KIVI	80.51 [77.66, 83.02]	82.14 [76.57, 87.37]	80.84 [78.22, 83.19]	-1.054 [-2.480, +0.707]
System + QVG	78.54 [75.90, 81.16]	82.07 [76.63, 87.34]	79.25 [76.81, 81.64]	-2.646 [-3.262, -2.050]
System + RIQ	81.93 [79.13, 84.53]	84.20 [78.65, 89.35]	82.38 [79.80, 84.80]	+0.487 [-0.237, +1.760]
OSR + RIQ	81.21 [78.48, 83.82]	83.53 [78.05, 88.74]	81.68 [79.23, 84.06]	-0.216 [-0.405, -0.043]
I2V Method	Quality↑	Alignment↑	Total↑	ΔTotal
Dense	78.24 [77.21, 79.32]	96.35 [95.13, 97.47]	87.30 [86.48, 88.06]	–
System	78.22 [77.18, 79.29]	96.34 [95.11, 97.46]	87.28 [86.47, 88.04]	-0.018 [-0.042, +0.006]
OSR	78.52 [77.45, 79.64]	96.40 [95.17, 97.51]	87.46 [86.65, 88.24]	+0.162 [-0.124, +0.493]
OSR + KIVI	73.93 [72.85, 75.04]	91.54 [89.95, 93.01]	82.74 [81.69, 83.72]	-4.562 [-5.419, -3.788]
OSR + QVG	73.76 [72.65, 74.87]	86.43 [84.55, 88.23]	80.09 [78.94, 81.23]	-7.206 [-8.263, -6.237]
System + KIVI	73.56 [72.41, 74.70]	90.24 [88.65, 91.77]	81.90 [80.84, 82.90]	-5.398 [-6.296, -4.582]
System + QVG	73.29 [72.22, 74.36]	86.44 [84.56, 88.21]	79.87 [78.69, 80.98]	-7.433 [-8.454, -6.497]
System + RIQ	78.13 [77.11, 79.17]	96.29 [95.06, 97.38]	87.21 [86.42, 87.96]	-0.091 [-0.391, +0.209]
OSR + RIQ	77.82 [76.79, 78.86]	96.40 [95.17, 97.49]	87.11 [86.33, 87.86]	-0.191 [-0.542, +0.146]

D QUALITATIVE EXAMPLES

Figures 8–11 provide selected qualitative comparisons for each workload and task. Every pair uses the same input and five uniformly spaced, temporally aligned keyframes; identifiers are retained in the artifact.

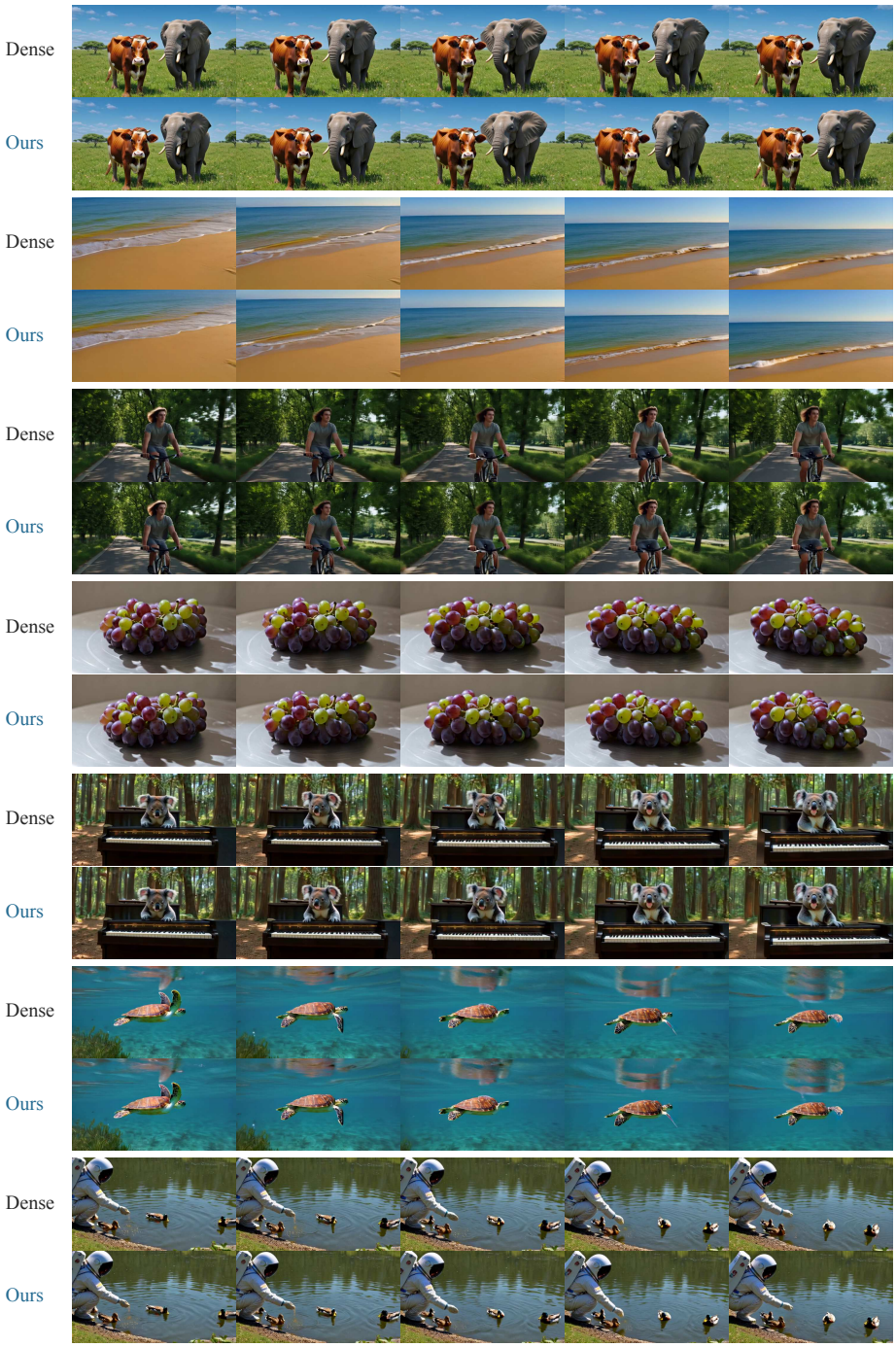


Figure 8: **720p/5s T2V.** Seven selected examples comparing Dense and OSR with FFN FP8 (Ours). Each pair contains five aligned keyframes.

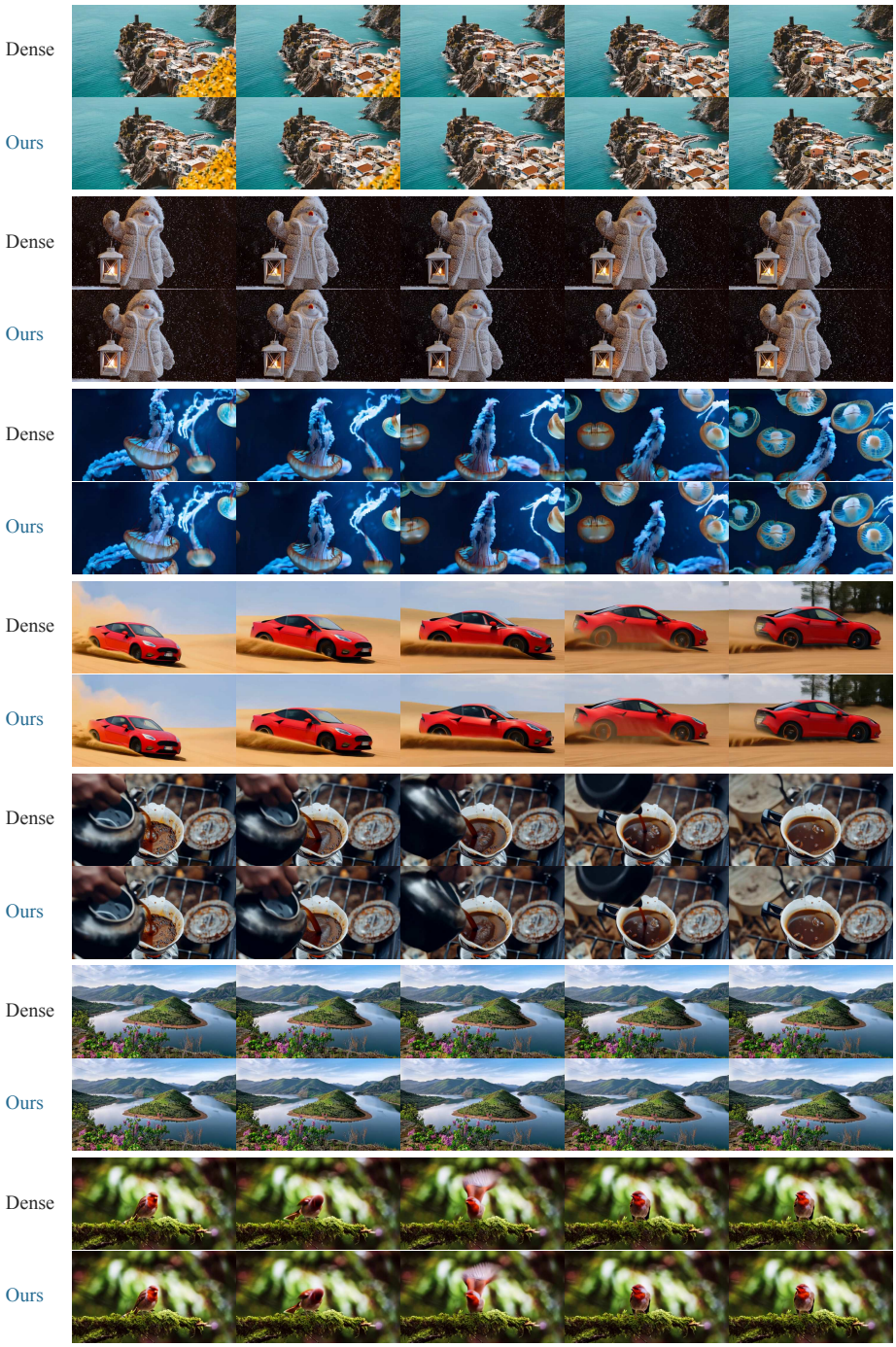


Figure 9: **720p/5s I2V.** Seven selected examples comparing Dense and OSR with FFN FP8 (Ours). Each pair contains five aligned keyframes.



Figure 10: **480p/10s T2V.** Seven selected examples comparing Dense and OSR with RIQ (Ours). Each pair contains five aligned keyframes.

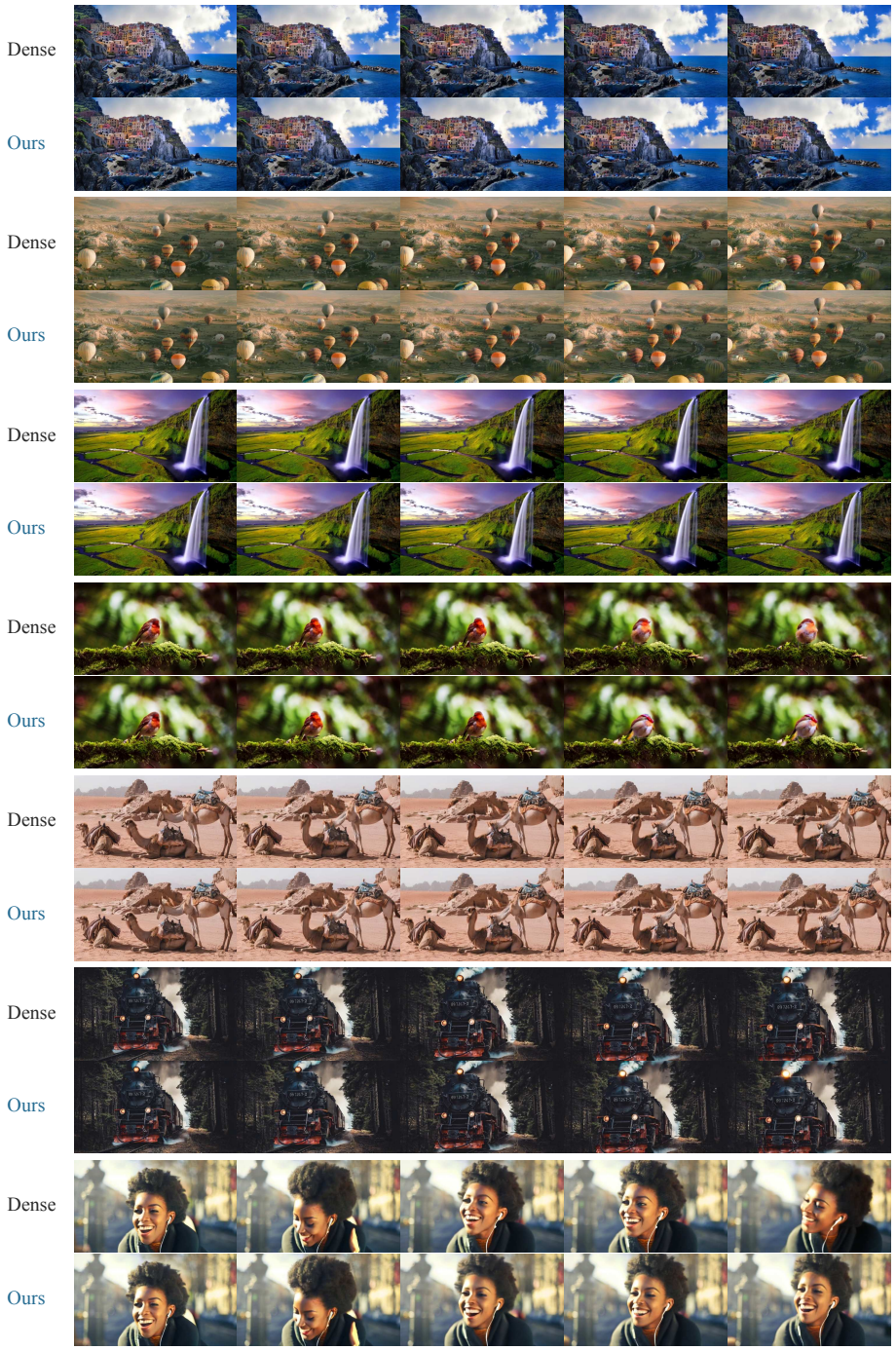


Figure 11: **480p/10s I2V.** Seven selected examples comparing Dense and OSR with RIQ (Ours). Each pair contains five aligned keyframes.