SUBITKV: SUB-BIT KV CACHE COMPRESSION FOR AUTOREGRESSIVE VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive video generation relies on a growing key–value (KV) cache to retain information from previous frames, which raises a memory challenge as video length increases. However, retaining the full KV cache at high precision becomes impractical under limited GPU memory. Discarding past states or reducing their precision saves memory but can compromise consistency in appearance, structure, and motion over time. To address this challenge, we introduce SuBitKV, a training-free KV cache compression framework that supports storage below one bit per original KV element. Its key idea is to preserve a low-memory approximation of the retained KV cache as a persistent base, then allocate additional bits across the KV cache within the memory budget to improve reconstruction fidelity. We construct this persistent base using residual vector quantization, which jointly represents the elements of each key or value vector with a small set of compact indices rather than storing each element separately. As the cache grows, SuBitKV reduces these extra bits to stay within budget while retaining the persistent base. Experiments on three video-generation and world-model backbones show a strong memory–quality trade-off. On Wan2.2, SuBitKV achieves $21.42\times$ compression with near-BF16 VBench scores and reaches $30.09\times$ under tighter memory budgets.

1 INTRODUCTION

Recent autoregressive video generators and interactive world models increasingly target long, continuous visual sequences (Team Wan et al., 2025; Sun et al., 2025; Huang et al., 2025). Maintaining coherence requires access to prior states that preserve subjects, scene structure, and motion. These states are stored in the transformer key–value (KV) cache, which serves as visual working memory during generation. As resolution and video length increase, the cache can become a dominant memory cost. Discarding cached states removes their availability to later attention; KV-cache quantization instead uses fewer bits per value while retaining the history.

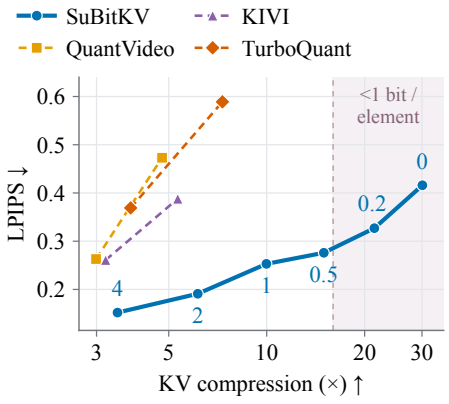


Figure 1: **Wan2.2 quality–compression trade-off.** LPIPS is measured against BF16; labels give SuBitKV's refinement budget Q. Shading indicates sub-bit storage.

Existing quantization methods such as KIVI and QuantVideo reduce KV-cache storage to 2 bits per element (Liu et al., 2024; Xi et al., 2026), substantially lowering the cost of retaining history. However, their representations still assign a separate fixed-width code to each cached value. Under a tighter cache budget, this representation reaches a hard limit: once fewer than one bit is available per original KV element on average, even a one-bit code per value exceeds the budget. This makes sub-bit KV-cache compression a distinct representation challenge: how can the full KV cache remain represented when fewer than one bit is available per original element?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Wan2.2, 1280×704, 173 frames. Ours: 0.2-bit KV cache (21.42×); QuantVideo: 2-bit (4.77×).

(b) Temporal warping error (TWE).

Figure 2: **Delayed post-boundary discontinuity on Wan2.2.** QuantVideo INT2 ($4.77\times$) versus SuBitKV at $Q = 0.2$ ($21.42\times$). TWE denotes temporal warping error, which is motion-aligned frame error after optical-flow alignment (lower is better); error-map multipliers are relative to BF16. Dashed lines mark chunk boundaries.

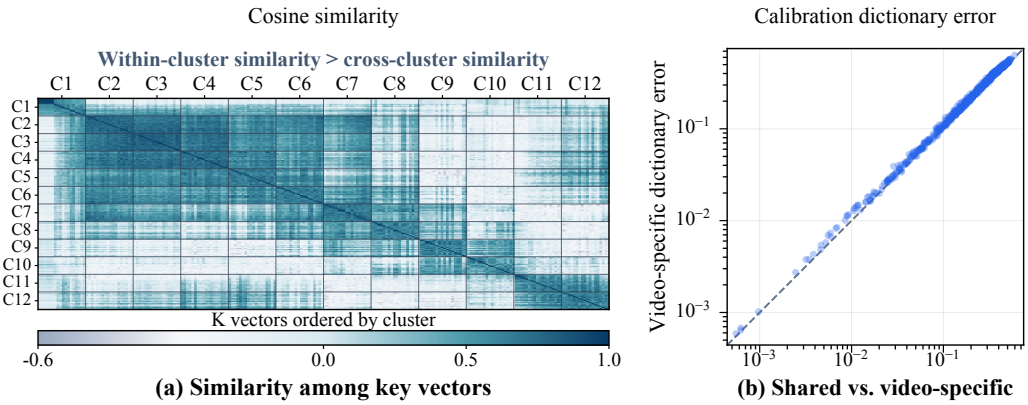
Therefore, we propose **SuBitKV**, a KV-cache compression framework for autoregressive video generation that compresses KV cache below one bit per element while preserving the full visual history. To achieve sub-bit storage, SuBitKV constructs a compact persistent base by jointly representing the elements of each key or value vector with shared codebook indices. We implement this base using residual vector quantization (RVQ), preserving an approximation of the full KV history. SuBitKV uses the remaining capacity within a fixed memory budget to store truncatable residual refinements, improving KV fidelity without increasing peak memory usage. It dynamically allocates refinement bits across chunks, retaining higher precision early in generation and progressively reducing it as the cache grows, without re-encoding. Even when a chunk's residual refinement is removed entirely, its persistent base remains available to attention.

We evaluate SuBitKV on three autoregressive video generation and world-model backbones: Wan2.2-5B, Self-Forcing-Wan-1.3B, and HY-WorldPlay 8B. Across models and memory budgets, the results demonstrate the effectiveness of sub-bit KV-cache compression. On Wan2.2, SuBitKV offers a strong memory–quality trade-off (Fig. 1), achieving $21.42\times$ KV-cache compression with near-BF16 VBench performance and reaching $30.09\times$ under tighter memory budgets, while incurring less than 3.4% end-to-end latency overhead. The example in Fig. 2 shows less delayed post-boundary discontinuity for SuBitKV at $21.42\times$ compression than QuantVideo INT2 at $4.77\times$. Beyond reducing cache storage, these savings enable HY-WorldPlay 8B to generate 189-frame videos on a single 24GB RTX 4090 with negligible quality degradation. SuBitKV lowers the memory barrier to long autoregressive video generation while preserving access to the full visual history.

2 Background and Motivation

KV-cache compression. Autoregressive video generators and world models use the KV cache to condition new frames on earlier content (Huang et al., 2025; Yang et al., 2026; Zhu et al., 2025). Rolling windows and selective caching control memory by restricting the history available to attention (Ma et al., 2026; Ji et al., 2026). WorldPlay reconstructs historical context to support geometric consistency (Sun et al., 2025). Another line of work compresses the retained KV cache. KIVI adapts quantization to key–value statistics (Liu et al., 2024), while TurboQuant combines random rotations with quantization (Zandieh et al., 2025). For video generation, QuantVideo uses chunk-specific centroids and residual refinement (Xi et al., 2026). These methods have advanced low-bit KV-cache compression. Our focus is on reducing storage further, to less than one bit per element on average while retaining the full KV history. This goal requires preserving temporal continuity alongside individual-frame quality. In our Wan2.2 comparison, for example, the QuantVideo INT2 output in Fig. 2 shows an appearance jump between the first two frames after a chunk boundary despite visually plausible individual frames. We therefore examine whether recurring structure in the KV cache can support a more compact representation while preserving the information needed for continuous generation.

Shared vector structure. Vector quantization jointly represents the elements of a KV vector using compact codebook indices (Kumar, 2024). This approach can reduce storage per element, but its accuracy depends on whether a small dictionary captures the vectors encountered during generation. Figure 3(a) shows stronger within-cluster directional similarity, suggesting recurring



(a) Similarity among key vectors **(b) Shared vs. video-specific**

Figure 3: **Key similarity and dictionary reuse.** (a) Pairwise cosine similarity of key vectors ordered by k-means clusters, labeled C_1–C_{12}. (b) Reconstruction error on the same video using shared calibration and video-specific dictionaries; the dashed diagonal marks equal error.

patterns that compact codebook indices can represent. Panel (b) tests reuse: a calibration dictionary gives reconstruction error comparable to a video-specific dictionary on the same video. This motivates dictionaries shared across videos, but separate for each layer, attention head, and tensor type. The remaining reconstruction error motivates allocating additional precision within the memory budget.

Progressive precision. Progressive quantization improves memory utilization by adapting precision as the KV cache grows. PM-KVQ progressively lowers KV bit-widths and assigns higher precision to more sensitive transformer blocks (Liu et al., 2025), allowing the smaller cache early in generation to use otherwise idle memory. Its precision schedule operates on per-element codes with bit-widths from 16 down to 2. Embedded image coding and nested quantization encode information at multiple precision levels, allowing refinement bits to be discarded to reduce storage (Said and Pearlman, 1996; Skodras et al., 2001; Nair et al., 2025). In SuBitKV, refinement bits encode the residual relative to a persistent sub-bit base, which remains available to attention when residual precision reaches zero. This lets us use spare memory to improve reconstruction accuracy and later reclaim it without removing historical KV vectors.

3 METHOD

Based on the insights in Section 2, we introduce SuBitKV, a sub-bit KV-cache compression framework for autoregressive video generation. Figure 4 illustrates the dataflow of SuBitKV. For each cached key or value vector $\mathbf{x}$, let $\mathbf{s}$ denote its *persistent base*: a fixed approximation of $\mathbf{x}$ with the same dimension. Stage 1 constructs $\mathbf{s}$ by summing entries selected from offline-shared codebooks $\{\mathcal{C}_l\}$ and stores their indices $\mathbf{z}$, rather than an additional dense copy of $\mathbf{s}$. Stage 2 encodes the remaining residual $\mathbf{r} = \mathbf{x} - \mathbf{s}$ as a truncatable refinement. At chunk-admission step t, the precision scheduler assigns each cached chunk $i \leq t$ a nominal residual precision $b_i(t)$ based on sensitivity and age, subject to the memory budget. This precision controls both K and V refinements. As new chunks enter the cache, earlier refinements are truncated as needed without re-encoding, while the stored indices and corresponding bases remain unchanged. When attention reads the cache, codebook lookup and summation recover $\mathbf{s}$, and the decoded residual refinement is added to reconstruct $\mathbf{x}$ approximately.

3.1 DEPTH-ADAPTIVE PERSISTENT BASE

To store the full KV history at a sub-bit rate, we cannot assign quantization bits to each KV element independently: even one bit per element already exceeds the target rate. Figure 3 shows that KV vectors share similar patterns and that dictionaries learned from calibration videos can be reused across videos. This motivates us to represent each complete KV vector jointly using indices into shared dictionaries, spreading the index cost across all its elements. However, adding more stages improves reconstruction but also increases the indices that must be stored throughout generation. We thus propose a **depth-adaptive persistent base** that keeps adding stages only when they provide enough reconstruction gain.

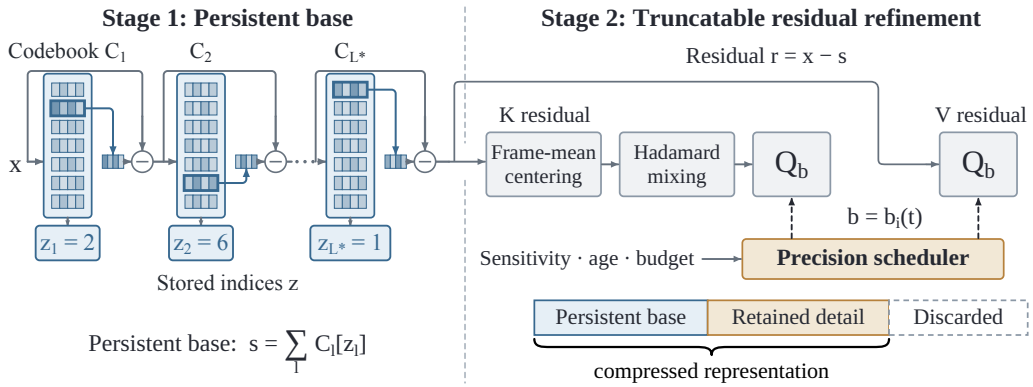


Figure 4: **SuBitKV overview.** Stage 1 stores indices $\mathbf{z}$ into shared codebooks whose selected entries sum to the persistent base $\mathbf{s}$. Stage 2 encodes $\mathbf{r} = \mathbf{x} - \mathbf{s}$ as truncatable refinements, with key-only conditioning. Dashed arrows indicate the scheduler-assigned precision $b = b_i(t)$ for both quantization paths. The brace groups the persistent base and retained refinement into the compressed representation, excluding discarded bits. Attention reconstructs $\hat{\mathbf{x}}_b = \mathbf{s} + Q_b(\mathbf{r})$, where Q_b decodes the retained refinement and $Q_0 = 0$.

We jointly encode each key or value across its full channel dimension within an attention head, without channel partitioning. Each dictionary entry, called a prototype, has the same dimension as the vector; selected prototypes are summed to approximate it, spreading their index cost across its elements. We learn the dictionaries offline from complete calibration vectors sampled separately for each layer, attention head, and tensor type. The first dictionary is formed by k-means centers, and subsequent dictionaries are learned from residuals left by preceding stages. The video generation model remains unchanged, and the dictionaries are fixed and reused across videos.

During generation, we construct the persistent base using residual vector quantization (RVQ) with the frozen dictionaries (Kumar, 2024). For a KV vector $\mathbf{x}$, let $\mathbf{r}^{(l-1)}$ denote the remaining error before stage l, starting with $\mathbf{r}^{(0)} = \mathbf{x}$. The stage selects a prototype from dictionary $\mathcal{C}_l = \{\mathbf{c}_k^{(l)}\}_{k=1}^{K}$:

$$z_l = \arg\min_{k \in \{1,\dots,K\}} \|\mathbf{r}^{(l-1)} - \mathbf{c}_k^{(l)}\|_2^2, \qquad \mathbf{r}^{(l)} = \mathbf{r}^{(l-1)} - \mathbf{c}_{z_l}^{(l)}. \tag{1}$$

After the selected number of stages $L^\star$, the stored indices reconstruct the persistent base as $\mathbf{s} = \sum_{l=1}^{L^\star} \mathbf{c}_{z_l}^{(l)}$. Subsequent attention reads reuse these indices without repeating the prototype selection.

To select the retained depth, we measure each stage's reduction in residual MSE as a fraction of the previous stage's MSE. Each stage adds a nominal index rate of $R_s = \log_2 K/d$ bits per element, where d is the full per-head vector dimension. Using the Gaussian rate–distortion relation as a reference, we set the stopping threshold to

$$\tau = 1 - \lambda 2^{-2R_s}, \tag{2}$$

where λ scales the Gaussian remaining-distortion reference 2^{-2R_s}. We stop when the relative MSE reduction falls below τ for two consecutive stages, subject to preset minimum and maximum stage counts. For each new chunk, we choose the depth for keys and values separately in each layer. The selected depth and indices then remain fixed, so later precision changes modify only the residual refinement while retaining the same persistent base.

3.2 KV-Aware Truncatable Refinement

Building on the compact persistent base in Section 3.1, we use the remaining memory budget to improve KV reconstruction by retaining residual detail. As new chunks enter the cache, more of the budget is needed for their bases, leaving less room for these residuals. This requires the stored residual representation itself to support precision reduction: independently quantized bit-widths would otherwise require historical residuals to be re-encoded. We therefore propose **KV-aware truncatable refinement**, which reconstructs lower precision directly from shorter prefixes of the stored codes:

$$\hat{\mathbf{x}}_b = \underbrace{\mathbf{s}}_{\text{persistent base}} + \underbrace{Q_b(\mathbf{x} - \mathbf{s})}_{\text{truncatable refinement}}. \tag{3}$$

Here, $\mathbf{x}$ is the original KV vector, $\mathbf{s}$ its persistent base defined in Section 3.1, and $\hat{\mathbf{x}}_b$ its reconstruction at nominal residual precision b. The decoder Q_b uses Q_b^K for keys and Q_b^V for values, with $Q_0 = 0$ for base-only reconstruction.

To recover lower precision directly from stored data, each retained prefix identifies a coarser interval of the original quantization grid. For each complete residual vector, we store a positive scale a. Let b_0 denote the initial code bit-width. For a value residual element r, the initial code is $c = q - q_{\min}$, where q is obtained by rounding and clipping r/a to 2^{b_0} consecutive integer levels starting at $q_{\min}$. Thus, $c \in \{0, \ldots, 2^{b_0} - 1\}$. Retaining the first b most significant bits gives $p_b = \lfloor c/s_b \rfloor$, where $s_b = 2^{b_0 - b}$. This prefix identifies s_b adjacent grid levels, whose midpoint gives the reconstruction:

$$Q_b^V(r) = a\left(q_{\min} + s_b p_b + \frac{s_b - 1}{2}\right), \qquad 1 \le b \le b_0. \tag{4}$$

Lower-precision reconstructions therefore reuse the same stored code and scale, without a new quantization pass.

Truncatability alone does not address the different residual statistics of K and V. Figure 5 shows that key residuals generally have more uneven channel energy than value residuals. With one scale per vector, channels with different energy levels share the same quantization step. We therefore condition keys before applying the prefix quantizer: we subtract the spatial mean within each latent frame and apply a signed Hadamard transform over the full per-head channel dimension to spread the remaining energy more evenly (Ashkboos et al., 2024). Value residuals use the prefix quantizer directly.

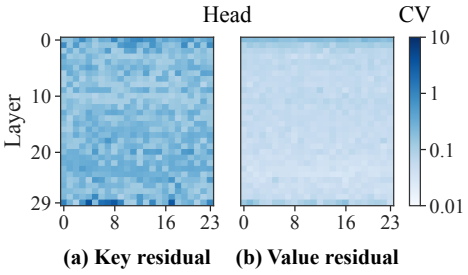


Figure 5: **Base-residual energy variation.** Each cell shows the channel-energy CV (standard deviation / mean) for one layer and head. Darker colors indicate greater unevenness on a shared logarithmic scale.

For a fixed layer and attention head, let $\mathbf{r}_{K,j}$ be the base residual of spatial token j in latent frame f, $\widetilde{\boldsymbol{\mu}}_f$ the stored per-channel mean over that frame's spatial tokens, and $\mathbf{R}$ the orthogonal signed Hadamard transform. The transformed residual $\mathbf{y}_{K,j}$ and its reconstruction are

$$\begin{aligned} \mathbf{y}_{K,j} &= \mathbf{R}(\mathbf{r}_{K,j} - \widetilde{\boldsymbol{\mu}}_f), \\ Q_b^K(\mathbf{r}_{K,j}) &= \widetilde{\boldsymbol{\mu}}_f + \mathbf{R}^\top Q_b^V(\mathbf{y}_{K,j}), \qquad 1 \le b \le b_0. \end{aligned} \tag{5}$$

Here, Q_b^V acts elementwise using the transformed key vector's own scale, grid parameters, and code prefixes; the superscript V identifies the shared quantizer, not its input type. The transpose $\mathbf{R}^\top$ reverses the transform, and adding $\widetilde{\boldsymbol{\mu}}_f$ restores the mean.

The scale and, for K, the frame mean remain fixed across positive-precision prefixes. Reducing b therefore reuses the stored representation without refitting these fields or recomputing the persistent base. At $b = 0$, we set $Q_0^K = Q_0^V = 0$ and remove all residual codes, scales, and K means. Refinement storage can thus be reclaimed completely while the historical KV state remains represented by its unchanged persistent base.

3.3 Time-Adaptive Refinement Allocation

Truncatable refinement allows precision to be reduced without re-encoding; the remaining question is how much refinement each chunk should retain as the cache grows. Early in generation, only a few chunks occupy the fixed cache budget, leaving room for finer refinements. As new chunks arrive, their persistent bases consume more memory, so this extra detail cannot always be retained. We therefore propose time-adaptive refinement allocation, which uses the available capacity to improve KV reconstruction and selectively reduces residual precision as history grows, while preserving every chunk's persistent base.

A planning horizon of N historical chunks is specified before generation. For equally sized chunks, let M_{BF16} be one chunk's BF16 KV storage and M_{base} the chunk-local storage reserved for its

persistent base. Let M_{codebook} be the shared codebook storage, including codebook scales, counted once across all chunks. With an average refinement-bit allowance Q per KV element, we set

$$B = N\left(M_{\text{base}} + \frac{Q}{16} M_{BF16}\right) + M_{\text{codebook}}. \tag{6}$$

This budget B applies at every chunk admission. Figure 8(b) illustrates how capacity used for early refinements gives way to additional persistent bases. Panel (a) shows the corresponding precision reductions, including zero refinement without removing the cached chunk.

Uniform precision does not account for differences in how cached chunks influence subsequent attention. We therefore use offline replay on calibration videos to estimate chunk importance $w_i(t)$ from attention received at different ages, and distortion $D(b)$ from changes in attention distributions and outputs. Moreover, discarded refinement bits are no longer available, so later allocations are constrained by the precision still retained for each chunk.

At admission t ($1 \le t \le N$), the cache contains t historical chunks, with allocation $\mathbf{b}(t) = (b_1(t), \ldots, b_t(t))$. Here, $b_i(t)$ denotes the nominal precision retained for chunk i, selected from $\mathcal{B} = \{16, 8, 4, 3, 2, 1, 0\}$. We choose an allocation that minimizes calibrated attention-weighted distortion, subject to the cache budget and nonincreasing precision for resident chunks:

$$\begin{aligned} \mathbf{b}^\star(t) = &\underset{b_i(t)\in\mathcal{B},\, i=1,\ldots,t}{\arg\min} \sum_{i=1}^{t} w_i(t)\, D(b_i(t)) \\ \text{s.t.} \quad & M_{\text{codebook}} + \sum\nolimits_{i=1}^{t} \left[M_{\text{base},i}(t) + M_{\text{ref},i}(b_i(t))\right] \le B, \quad b_i(t) \le b_i(t-1) \quad (i < t). \end{aligned} \tag{7}$$

Here, $\mathbf{b}^\star(t)$ is the selected per-chunk residual-bit allocation, and $M_{\text{base},i}(t)$ accounts for chunk i's local persistent-base storage; shared codebooks are counted once in M_{codebook}. $M_{\text{ref},i}(b)$ counts residual codes, scales, and means, or BF16 KV at $b = 16$; $b = 0$ retains the persistent base only, with $M_{\text{ref},i}(0) = 0$.

Offline calibration is performed once per backbone. For the chosen planning horizon and cache budget, the calibrated importance and distortion profiles are fixed inputs to offline dynamic programming. The resulting allocations are stored in a table indexed by t and reused across prompts, so generation follows the schedule without rerunning dynamic programming. Generation may stop at any earlier step $T \le N$ by using the first T rows under the same budget B, without recalibration or rescheduling. Appendix A.1.1 details the procedure.

4 Experiments

4.1 Experimental Setup

Models and generation settings. We evaluate Wan2.2-5B (Team Wan et al., 2025) for text-to-video generation, HY-WorldPlay 8B (Sun et al., 2025) for world generation conditioned on an image, text, and camera actions, and Self-Forcing-Wan-1.3B (Huang et al., 2025) for video generation (Appendix A.4). Wan2.2 generates 11 chunks (173 decoded frames) at 1280×704; HY-WorldPlay follows its public 12-chunk image-conditioned protocol at 832×480; Self-Forcing generates 10 chunks at 832×480, with 12 decoded frames per subsequent chunk and a shorter initial chunk. Wan2.2 uses 50 denoising steps with guidance scale 1; Self-Forcing uses the four-step DMD sampler with EMA weights. Appendix A.2 specifies the evaluation and measurement scopes.

Evaluation data. We use MovieGenVideoBench (Polyak et al., 2024). Calibration uses 20 prompts unless otherwise stated; Figure 10 reports calibration-size sensitivity. Excluding the 20 calibration prompts and 10 prompts reserved for earlier experiments from the 1003-prompt benchmark leaves a 973-sample held-out split shared by all three models. For HY-WorldPlay, each text prompt is paired with the first frame of the corresponding MovieGen video as its image condition.

Metrics. PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) compare against same-seed BF16 generations, excluding uncompressed prefixes. VBench (Huang et al., 2024) scores (0–1 scale) cover subject consistency, background consistency, imaging quality, and aesthetic quality.

Delayed post-boundary discontinuity. Following motion-warped temporal evaluation (Lai et al., 2018), we use Temporal Warping Error (TWE) to measure the second-frame appearance jump after

Table 1: **KV-cache compression on Wan2.2 and HY-WorldPlay.** Quality scores use 973 samples; TWE uses 10 videos per setting. Configurations denote nominal quantization settings for baselines and the residual-budget parameter Q for SuBitKV. Compression ratios include shared codebooks, metadata, and cache-related workspace. Change is the percentage change in TWE from BF16; negative values indicate lower error. Dashes in BF16 reconstruction columns omit self-comparisons because BF16 is the reference.

			Reconstruction			VBench				Temporal	
Method	Config.	Comp. ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Subj. ↑	Bkg. ↑	Img. ↑	Aes. ↑	TWE ↓	Change ↓
Wan2.2-5B											
BF16	BF16	1.00×	–	–	–	.866	.897	.757	.562	0.0173	0.0%
QuantVideo	INT4	2.99×	19.59	.684	.263	.867	.897	.758	.562	0.0171	−1.4%
KIVI	INT4	3.20×	19.73	.689	.261	.867	.897	.757	.562	0.0172	−0.7%
TurboQuant	K4/V4	3.82×	15.92	.538	.369	.868	.898	.756	.565	0.0182	+4.9%
SuBitKV	$Q = 4$	**3.48**×	**27.60**	**.851**	**.152**	.866	.897	.757	.562	0.0171	−1.2%
SuBitKV	$Q = 2$	**6.14**×	**24.09**	**.793**	**.191**	.866	.897	.757	.562	0.0175	+0.6%
SuBitKV	$Q = 1$	**9.98**×	**20.82**	**.703**	**.253**	.865	.896	.756	.562	0.0180	+3.5%
QuantVideo	INT2	4.77×	13.94	.435	.473	.853	.888	.738	.562	0.0230	+32.4%
KIVI	INT2	5.33×	15.24	.517	.388	.867	.897	.752	.560	0.0193	+11.3%
TurboQuant	K2/V2	7.31×	12.88	.354	.589	.798	.863	.642	.494	0.0264	+52.0%
SuBitKV	$Q = 0.5$	**14.98**×	**20.19**	**.673**	**.276**	.864	.895	.754	.563	0.0190	+9.5%
SuBitKV	$Q = 0.2$	**21.42**×	**18.32**	**.610**	**.327**	.859	.892	.752	.563	0.0194	+12.1%
SuBitKV	$Q = 0$	**30.09**×	**16.30**	**.513**	**.416**	.841	.883	.754	.556	0.0244	+40.7%
HY-WorldPlay 8B											
BF16	BF16	1.00×	–	–	–	.852	.918	.594	.526	0.00798	0.0%
QuantVideo	INT4	3.30×	26.53	.823	.201	.852	.918	.592	.524	0.00813	+1.9%
KIVI	INT4	3.19×	26.58	.824	.201	.852	.919	.594	.525	0.00893	+12.0%
TurboQuant	K4/V4	3.82×	22.49	.724	.269	.852	.918	.594	.524	0.00947	+18.7%
SuBitKV	$Q = 4$	**3.41**×	**40.16**	**.907**	**.132**	.852	.918	.594	.525	0.00789	−1.1%
SuBitKV	$Q = 2$	**5.94**×	**31.07**	**.876**	**.161**	.852	.918	.593	.525	0.00799	+0.2%
SuBitKV	$Q = 1$	**9.45**×	**26.31**	**.799**	**.220**	.847	.916	.587	.522	0.00879	+10.1%
QuantVideo	INT2	5.61×	18.87	.574	.408	.822	.901	.567	.493	0.0194	+143.7%
KIVI	INT2	5.32×	20.13	.650	.341	.849	.915	.576	.509	0.00989	+24.0%
TurboQuant	K2/V2	7.31×	16.45	.485	.498	.788	.888	.498	.440	0.0256	+221.2%
SuBitKV	$Q = 0.5$	**13.33**×	**25.11**	**.759**	**.252**	.841	.913	.585	.519	0.0106	+32.5%
SuBitKV	$Q = 0.2$	**18.20**×	**22.93**	**.698**	**.305**	.832	.908	.582	.516	0.0125	+56.8%
SuBitKV	$Q = 0$	**24.02**×	**21.43**	**.655**	**.341**	.827	.906	.580	.520	0.0202	+153.3%

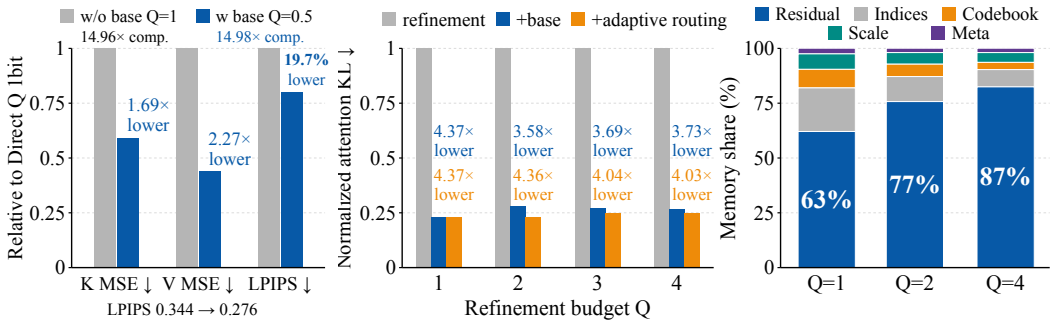
each chunk boundary. We evaluate the fixed $(e_k + 1, e_k + 2)$ transition without window maximization and average results over 10 videos per setting. The tables also report percentage change from BF16; Appendix A.3 gives the full protocol.

Baselines. QuantVideo (Xi et al., 2026), TurboQuant (Zandieh et al., 2025), and KIVI (Liu et al., 2024) adapters share a common retention policy, prompts, seeds, and generation settings; each admitted history chunk is quantized once. Table 5 summarizes the common evaluation protocol. Throughout the evaluation, cache accounting covers dynamic KV, shared codebooks, scales, predictors, cache metadata, and peak cache-related quantization/dequantization workspace (Appendix A.2).

Implementation. Compression precedes positional encoding. We use layer–head-specific 64-entry INT4 RVQ codebooks over the full per-head vector, with adaptive depth from 4–16 stages using Eq. (2). Q denotes the residual-bit allowance rather than the effective cache-memory rate. Layer–head precision routing is determined offline from calibration statistics. Appendix A.1 and Appendix A.2 give the codec, calibration, schedule-reuse, and hardware details.

4.2 Main Results

Sub-bit storage with improved reference fidelity. On Wan2.2, Q=0.2 reaches $21.42\times$ compression, or 0.747 effective bits per KV element ($b_{\text{eff}} = 16/R$). Compared with KIVI INT2, it uses a $4.02\times$ smaller cache and reduces LPIPS from 0.388 to 0.327 (15.7%). VBench remains close to BF16 on Wan2.2 and HY-WorldPlay (Table 1). At Q=0.5 on Wan2.2, SuBitKV reaches $14.98\times$ compression with 0.276 LPIPS, versus $4.77\times$ and 0.473 for QuantVideo INT2: a $3.14\times$ smaller cache with 41.6% lower LPIPS. Even at Q=0, Wan2.2 reaches $30.09\times$ compression with 0.416 LPIPS, still 12.1% lower than QuantVideo INT2.



(a) Local error and video LPIPS **(b) Attention preservation** **(c) Memory composition**

Figure 6: **Representation quality and storage.** (a) K/V MSE and video LPIPS with ($Q = 0.5$) and without ($Q = 1$) the persistent base, normalized to the latter. LPIPS uses default temporal scheduling. (b) Attention KL relative to BF16, normalized to refinement-only at each Q. Refinement denotes direct quantization; +base adds the persistent base; +adaptive routing adds fixed, offline-calibrated layer–head precision allocation. (c) Memory composition at $Q = 1, 2, 4$.

Higher fidelity across backbones. On Wan2.2, Q=2 gives $6.14\times$ compression with 0.191 LPIPS, using a $1.92\times$ smaller cache than KIVI INT4 while lowering LPIPS by 26.8%. On HY-WorldPlay, Q=2 yields 31.07 dB/0.161 at $5.94\times$ compression, versus approximately 26.5 dB/0.201 for 4-bit QuantVideo and KIVI at about $3.2\times$. Self-Forcing shows the same trend (Appendix A.4).

Delayed post-boundary discontinuity. At Q=1, SuBitKV lowers next-pair TWE relative to QuantVideo INT2 by 21.8% on Wan2.2 and 54.8% on HY-WorldPlay. Self-Forcing shows the same trend, although KIVI INT2 is lower; at Q=0, SuBitKV has higher TWE than QuantVideo INT2 on all three models (Appendix A.4).

Cross-metric trade-offs. At Q=0.2 on HY-WorldPlay, SuBitKV uses $3.42\times$ smaller cache footprints than KIVI INT2 and improves LPIPS from 0.341 to 0.305, while subject consistency decreases from 0.849 to 0.832 and next-pair TWE increases by 26.4% (0.009891 to 0.012505).

4.3 Representation Ablations

Persistent base. At nearly matched storage, Figure 6(a) shows that adding the persistent base lowers LPIPS from 0.344 to 0.276 (19.7%) and reduces K/V reconstruction MSE by $1.69\times$/$2.27\times$ relative to refinement-only quantization ($14.98\times$ versus $14.96\times$ compression). These gains at comparable cache sizes indicate that the joint base–refinement representation uses the available bits more effectively than refinement alone.

Hadamard conditioning for keys. Table 2 evaluates Hadamard mixing on the base residual. For K residuals, the transform reduces the channel-energy coefficient of variation by 89.2% and kurtosis by 88.9%; logit and attention-output MSE decrease by 70.0% and 94.5%. V residuals are already more isotropic, supporting the lighter direct path for V.

Table 2: **Effect of Hadamard mixing.**

Metric	w/o H.	w H.	Δ
K energy CV	.2551	.0276	−89.2%
K kurtosis	9.274	1.030	−88.9%
V energy CV	.0156	.0143	−8.1%
Logit MSE	1.50×10^{-2}	4.50×10^{-3}	−70.0%
Output MSE	1.39×10^{-4}	7.61×10^{-6}	−94.5%

Attention preservation and sensitivity allocation. The persistent base lowers attention KL by 3.58–$4.37\times$ relative to direct quantization (Figure 6(b)). At 2 bits, layer–head routing further reduces KL by approximately 17.9% relative to the same base–refinement representation without routing , reaching a $4.36\times$ total reduction. This diagnostic fixes average residual precision, with additional storage for the persistent base; panel (a) provides the comparison at nearly matched total storage.

4.4 Time-Adaptive Precision Allocation

Uniform Q=2 outperforms the compression baselines in per-chunk LPIPS, and time-adaptive Q=2 improves it further at the same nominal peak KV capacity (Figure 7(b)). Scheduled Q=1 and Q=2 retain Imaging Quality close to BF16 (Figure 7(a)).

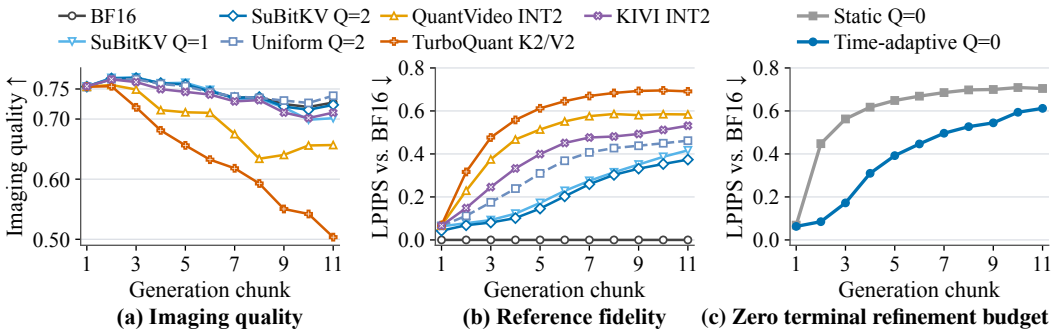


Figure 7: **Quality over generation on Wan2.2.** (a) Imaging Quality; (b) per-chunk LPIPS; (c) static $Q = 0$ uses no residual refinements throughout generation, while time-adaptive $Q = 0$ retains temporary refinements before removing them all at the final admission. LPIPS in (b,c) is evaluated independently per chunk. Uniform and time-adaptive Q=2 share the same nominal peak KV capacity.

Zero terminal residual budget. Figure 7(c) compares static and time-adaptive Q=0, both ending with zero residual payload. Time-adaptive precision uses otherwise idle cache capacity for temporary refinements and lowers independently evaluated per-chunk LPIPS, including on the final chunk. Early cache reads condition subsequent chunks, which in turn produce new keys and values. Earlier precision can therefore affect states later retained in a base-only cache; each run retains its own generated history. LPIPS rises with generation length in both settings, with lower deviation from BF16 under adaptive precision.

4.5 Efficiency Evaluation

Memory footprint and cache capacity. Figure 6(c) breaks down cache storage across refinement budgets. Lower budgets reclaim refinement storage while preserving the persistent bases; at $Q = 0$, the terminal refinement payload is fully removed.

Offline calibration cost. Offline calibration uses 20 videos per backbone to fit shared codebooks and routing profiles, taking approximately 62 minutes on Wan2.2 and 140 minutes on HY-WorldPlay, mostly for generation and KV-cache saving. This one-time cost is below 30 times the corresponding single-video end-to-end latency; fitted parameters are reused across held-out generations.

End-to-end latency. Table 3 reports mean Wan2.2 runtime. SuBitKV adds 0.44–3.37% end-to-end overhead relative to BF16 in these measurements. Encoding takes 1.26–1.59 s, while decoding increases from 2.00 s at $Q = 0$ to 5.13 s at $Q = 4$ as more residual detail is retained. These codec costs keep the runtime impact small across the reported budgets. Calibration and dynamic programming remain offline.

Table 3: Wan2.2 runtime (s) and overhead over BF16. Enc./Dec.: encoding/decoding.

Config.	E2E	Overhead	Enc. / Dec.
BF16	255.85	—	—
$Q = 0$	258.20	+0.92%	1.26 / 2.00
$Q = 0.2$	261.03	+2.02%	1.50 / 2.56
$Q = 0.5$	263.76	+3.09%	1.48 / 3.38
$Q = 1$	257.71	+0.73%	1.44 / 4.01
$Q = 2$	256.99	+0.44%	1.51 / 4.32
$Q = 4$	264.49	+3.37%	1.59 / 5.13

5 Conclusion

In this work, we introduce SuBitKV, a training-free sub-bit KV cache compression framework for autoregressive video generation. It uses residual vector quantization to maintain a low-memory approximation of the full KV cache. Our time-adaptive strategy further uses available memory to improve cache fidelity without increasing peak memory usage. Extensive experiments across three video-generation and world-model backbones demonstrate a strong memory–quality trade-off. On Wan2.2, SuBitKV achieves $21.42\times$ compression with near-BF16 VBench scores and reaches $30.09\times$ under tighter memory budgets. Together, these results demonstrate that sub-bit KV compression can make full-history autoregressive video generation practical under limited GPU memory.

Reproducibility Statement

Section 4.1 and Appendix A.2 report generation settings, evaluation protocols, hardware, and cache-memory accounting. Section 3 and Appendix A.1 describe the codec, calibration, and precision-schedule construction. Appendix A.3 specifies the temporal-error evaluation.

AI Use Statement

Generative AI tools were used to assist with manuscript drafting and editing, figure and table presentation, and feedback on research methodology, experimental design, and interpretation of results. The authors reviewed all AI-assisted work and take responsibility for the final content, claims, and artifacts in this submission.

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. *arXiv preprint arXiv:2404.00456*, 2024. URL https://arxiv.org/abs/2404.00456.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025. URL https://arxiv.org/abs/2506.08009.

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*, 2024. URL https://arxiv.org/abs/2311.17982.

Yicheng Ji, Zhizhou Zhong, Jun Zhang, Qin Yang, XiTai Jin, Ying Qin, Wenhan Luo, Shuiyang Mao, Wei Liu, and Huan Li. Forcing-kv: Hybrid kv cache compression for efficient autoregressive video diffusion models. *arXiv preprint arXiv:2605.09681*, 2026. URL https://arxiv.org/abs/2605.09681.

Ankur Kumar. Residual vector quantization for KV cache compression in large language model. In *Proceedings of the 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pages 485–490. PMLR, 2024. URL https://proceedings.mlr.press/v262/kumar24a.html.

Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *European Conference on Computer Vision*, 2018. URL https://arxiv.org/abs/1808.00449.

Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. PM-KVQ: Progressive mixed-precision KV cache quantization for long-CoT LLMs. *arXiv preprint arXiv:2505.18610*, 2025. URL https://arxiv.org/abs/2505.18610.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32332–32344. PMLR, 2024. URL https://proceedings.mlr.press/v235/liu24bz.html.

Yuexiao Ma, Xuzhe Zheng, Jing Xu, Xiwei Xu, Feng Ling, Xiawu Zheng, Huafeng Kuang, Huixia Li, Xing Wang, Xuefeng Xiao, Fei Chao, and Rongrong Ji. Flow caching for autoregressive video generation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vko4DuhKbh.

Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1711.07837.

Pranav Ajit Nair, Puranjay Datta, Jeff Dean, Prateek Jain, and Aditya Kusupati. Matryoshka Quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 45484–45506, 2025. URL https://proceedings.mlr.press/v267/nair25a.html.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie Gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. URL https://arxiv.org/abs/2410.13720.

Amir Said and William A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3): 243–250, 1996. doi: 10.1109/76.499834.

Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001. doi: 10.1109/79.952804.

Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. WorldPlay: Towards long-term geometric consistency for real-time interactive world modeling. *arXiv preprint arXiv:2512.14614*, 2025. URL https://arxiv.org/abs/2512.14614.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, 2020. URL https://arxiv.org/abs/2003.12039.

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. URL https://ece.uwaterloo.ca/~z70wang/publications/ssim.html.

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. *arXiv preprint arXiv:2602.02958*, 2026. URL https://arxiv.org/abs/2602.02958.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. Longlive: Real-time interactive long video generation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.22622.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. *arXiv preprint arXiv:2504.19874*, 2025. URL https://arxiv.org/abs/2504.19874.

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*, 2018. URL https://arxiv.org/abs/1801.03924.

Yixuan Zhu, Jiaqi Feng, Wenzhao Zheng, Yuan Gao, Xin Tao, Pengfei Wan, Jie Zhou, and Jiwen Lu. Astra: General interactive world model with autoregressive denoising. *arXiv preprint arXiv:2512.08931*, 2025. URL https://arxiv.org/abs/2512.08931.

A Appendix

A.1 Method Configuration and Precision Schedule

Table 4 supplements Sections 3.1–3.3; Figure 8 shows the stored schedule used online.

Table 4: **Method configuration.** Settings are frozen before held-out generation.

Component	Setting
Encoding unit	One complete per-head K/V vector; no channel partitioning. Dictionaries are separate for each layer, head, and tensor type.
Persistent base (§3.1)	64 prototypes per RVQ stage, stored in INT4; 4–16 stages. Stop after two consecutive relative MSE reductions below τ in Eq. (2), with $\lambda = 1$. Depth and indices then remain fixed.
K refinement (§3.2)	Frame-wise spatial centering and full-vector signed Hadamard mixing, followed by truncatable residual refinement. The decoder reverses the transform and restores the stored mean.
V refinement (§3.2)	Truncatable refinement in the original residual coordinates. Positive-rate truncation reuses the stored scale; no historical residual is re-encoded.
Precision states	$b = 16$: BF16 KV; $b \in \{8, 4, 3, 2, 1\}$: persistent base plus residual codes and local metadata; $b = 0$: base only, with residual codes, scales, and K means removed.
Calibration	20 videos per backbone. BF16 replay supplies age-dependent attention-mass weights; K records attention KL and output normalized MSE, and V records output normalized MSE.
Allocation (§3.3)	DP uses calibrated importance and distortion profiles to construct a schedule for the planning horizon and cache budget. The stored schedule is reused across prompts.
Cache lifecycle	Construct each incoming persistent base once; look up target precisions; truncate old prefixes before installing new persistent state. Attention reads decode retained history; later truncations do not recompute the base or refit scales.

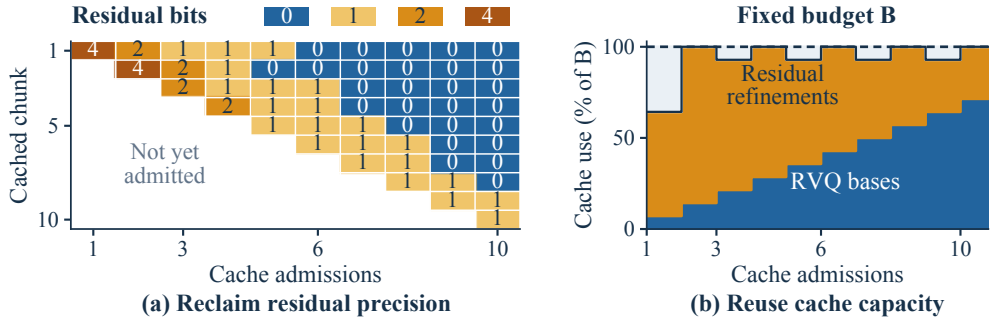


Figure 8: **Wan2.2** $Q = 0.2$ **precision schedule (Section 3.3).** (a) White cells indicate unadmitted chunks; zero bits retain the persistent base only. The first chunk follows $4 \to 2 \to 1 \to 0$; the final two retain one-bit refinements. In contrast, $Q = 0$ ends with no residuals. (b) Illustrative resident-cache allocation with equal-sized bases.

A.1.1 Schedule Construction and Execution

The procedure below expands Eq. (7) into offline schedule construction and online cache updates.

1. **Calibrate the allocation objective.** Replay the calibration videos to estimate age-dependent chunk importance $w_i(t)$ and the distortion $D(b)$ at each supported precision. Freeze these profiles before held-out generation; they provide the coefficients of the allocation objective.
2. **Set the horizon and budget.** Choose a planning horizon N and cache budget B as in Eq. (6). Construct one schedule row for each admission $t = 1, \ldots, N$. At each admission, reserve storage for the shared codebooks and all resident persistent bases; the remaining capacity is available for refinements.
3. **Solve the current allocation.** For each existing chunk, allow only precision states no greater than its allocation in the preceding row. The incoming chunk may use any state in $\mathcal{B}$. DP processes the resident chunks one at a time, recording the minimum accumulated weighted distortion for each storage allowance. Each transition assigns an admissible precision to the next chunk and adds its storage cost $M_{\mathrm{ref},i}(b)$ and distortion $w_i(t)D(b)$. Discard transitions that exceed the available capacity.
4. **Recover and store the schedule.** Backtrack through the minimizing transitions to recover $\mathbf{b}^\star(t)$. Store this row and use its precisions as upper bounds for existing chunks at the next admission. Repeat until all rows have been constructed, then reuse the table across prompts.
5. **Execute the schedule during generation.** At admission t, read the stored row, truncate older refinements to their assigned precisions, and install the incoming chunk in its selected state. Attention decodes the retained representations, including persistent bases when residuals have been removed. Calibration and DP are performed offline. A run ending at $T \leq N$ uses the first T rows and satisfies the same budget B at every admission.

A.2 Evaluation Settings and Memory Accounting

Table 5 supplements Section 4.1.

Table 5: **Evaluation settings and measurement scopes.**

Item	Protocol
Wan2.2-5B	1280×704, 16 fps; 11 chunks / 173 decoded frames; 50 denoising steps, guidance scale 1.
Self-Forcing	832×480; 10 chunks, 12 decoded frames per subsequent chunk and a shorter initial chunk; four-step DMD sampler with EMA weights.
HY-WorldPlay-8B	Image, text, and camera-action conditioning; 12 chunks for main quality evaluation, 10 chunks for the temporal subset.
Held-out split	973 MovieGenVideoBench samples, excluding 20 calibration prompts and 10 earlier-experiment prompts from 1003. HY uses each source video's first frame as the image condition.
Reconstruction	Same-seed BF16 references; uncompressed prefixes excluded. Self-Forcing VBench coverage is specified in Table 7; TWE uses 10 videos per setting.
Common baseline policy	Same prompts, seeds, settings, attention-visible history, and retention policy. Each admitted chunk is encoded once and decoded on later reads; no extra persistent high-precision history window.
Compression interface	Before positional encoding; decoded KV enters the original attention interface.
Precision units	$b_i(t)$: chunk i's retained precision; Q: target residual-budget parameter; $b_{\mathrm{eff}} = 16/R$: effective physical rate at measured compression ratio R.
Hardware	A100 GPUs for large-scale runs; RTX 6000 Ada GPUs for system profiling.

Cache-memory scope. Using the chunk-local terms of Eq. (7),

$$
\begin{aligned}
M_{\mathrm{persistent}}(t) &= M_{\mathrm{shared}} + M_{\mathrm{metadata}} + \sum_{i=1}^{t} \left[M_{\mathrm{base},i}(t) + M_{\mathrm{ref},i}(b_i(t)) \right], \\
M_{\mathrm{cache,peak}} &= \max_t \left[M_{\mathrm{persistent}}(t) + M_{\mathrm{workspace}}(t) \right].
\end{aligned}
\tag{8}
$$

Shared codebooks and frozen codec parameters are counted once; local refinement includes its required scales and means, with $M_{\mathrm{ref},i}(0) = 0$. All terms use physical stored bytes. Cache-related workspace is included; model weights and unrelated activations are not.

Historical chunks decode directly into the current layer's preallocated K/V buffers, reusing chunk-local scratch rather than a second concatenated history copy.

A.3 Delayed Post-Boundary Discontinuity Protocol

Table 6 specifies the targeted temporal metric in Section 4.1, also reported in Tables 1 and 7.

Table 6: **Delayed post-boundary discontinuity evaluation.** The offset is identical across methods.

Step	Setting
Subset	10 videos per setting for temporal evaluation; quality evaluation uses 973 samples. Prompts, seeds, and chunk boundaries match across methods within each backbone.
Frame selection	If chunk k ends at e_k, score exactly $(e_k + 1, e_k + 2)$ when both frames exist; no offset search or window maximization.
Optical flow	Resize RGB to 640×352; torchvision RAFT-small with official default weights and preprocessing (Teed and Deng, 2020). Estimate forward flow F and backward flow B.
Alignment and mask	Backward-warp I_t and F into frame $t+1$ coordinates using B, yielding $\widehat{I}_t$ and $\widehat{F}$. Mask M_t retains in-bounds pixels satisfying $\|B + \widehat{F}\|_2^2 \le 0.01(\|B\|_2^2 + \|\widehat{F}\|_2^2) + 0.5$ (Meister et al., 2018).
Signal	Luminance $Y = 0.2126R + 0.7152G + 0.0722B$; exclude empty-mask pairs rather than assign zero.
Aggregation	Average valid selected pairs within each video, then equally across videos. Report TWE and its percentage change from BF16; lower is better.

$$\mathrm{TWE}_{i,t}^{m} = \frac{\sum_x M_t(x)\,|Y(I_{t+1})(x) - Y(\widehat{I}_t)(x)|}{\sum_x M_t(x)}. \tag{9}$$

This fixed offset targets the observed second-frame appearance jump. TWE is reported alongside reconstruction and VBench scores.

A.4 Additional Backbone Results

Table 7 extends the main comparison (Section 4.2) to Self-Forcing.

Table 7: **Self-Forcing results.** Metrics and notation follow Table 1; reconstruction and VBench use the same 973 prompts for all methods and configurations. TWE uses 10 videos per setting.

			Reconstruction			VBench				Temporal	
Method	Config.	Comp. ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Subj. ↑	Bkg. ↑	Img. ↑	Aes. ↑	TWE ↓	Change (%) ↓
BF16	BF16	1.00×	–	–	–	.940	.944	.701	.603	0.0114	0.0%
QuantVideo	INT4	3.16×	25.02	.816	.102	.940	.944	.701	.603	0.0116	+1.3%
KIVI	INT4	3.19×	25.32	.823	.099	.940	.944	.701	.603	0.0115	+1.0%
TurboQuant	K4/V4	3.82×	19.31	.659	.203	.939	.943	.696	.600	0.0113	−1.2%
SuBitKV	$Q = 4$	**3.37**×	**33.40**	**.916**	**.044**	.940	.944	.701	.603	0.0114	−0.1%
SuBitKV	$Q = 2$	**5.83**×	**30.27**	**.888**	**.060**	.940	.944	.701	.603	0.0115	+1.2%
SuBitKV	$Q = 1$	**9.18**×	**26.32**	**.831**	**.093**	.939	.944	.701	.603	0.0113	−0.6%
QuantVideo	INT2	5.22×	17.95	.605	.251	.933	.941	.699	.597	0.0132	+15.4%
KIVI	INT2	5.32×	19.09	.662	.195	.942	.945	.700	.602	0.0110	−3.2%
TurboQuant	K2/V2	7.31×	15.97	.502	.463	.846	.903	.588	.524	0.0142	+24.4%
SuBitKV	$Q = 0.5$	**13.20**×	**25.28**	**.802**	**.114**	.936	.943	.699	.602	0.0114	+0.1%
SuBitKV	$Q = 0.2$	**17.95**×	**22.79**	**.741**	**.161**	.932	.941	.698	.601	0.0120	+4.8%
SuBitKV	$Q = 0$	**23.53**×	**19.70**	**.637**	**.255**	.915	.932	.696	.595	0.0132	+15.6%

A.5 Persistent-Base Depth and Calibration

These controlled Wan2.2 diagnostics examine base reconstruction, conditioning, and temporal importance on the stated held-out subsets, separately from the 973-sample comparison in Table 1.

Diminishing stage gains. Figure 9(a) reports relative reductions in aggregate mean MSE over 720 layer–head units. The first stage removes 79.78% of key MSE and 34.65% of value MSE; at stage 64, marginal gains fall to 3.35% and 3.24%. With 64 prototypes, 128 channels, and $\lambda = 1$, the reference threshold is $\tau \simeq 6.29\%$. The aggregate curves first cross it at stages 11 and 7. These crossings illustrate diminishing returns; selected depths follow the per-chunk, two-consecutive-stage rule.

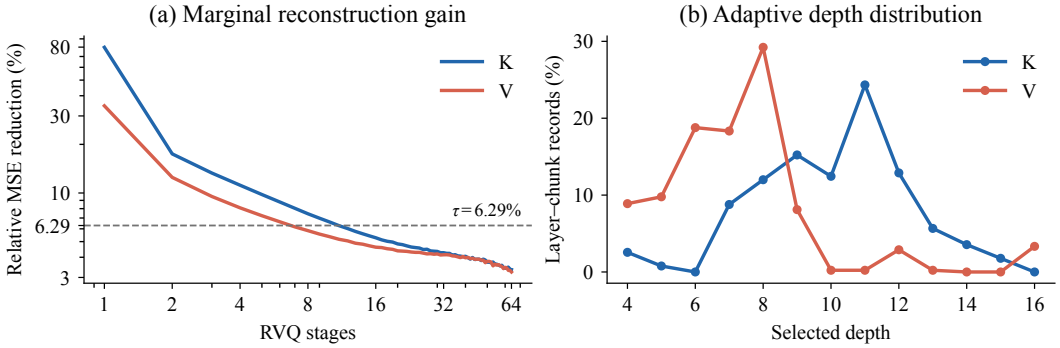


Figure 9: **Stage gains and selected depths.** Left: marginal MSE reduction from a 64-stage sweep, aggregated over 720 held-out layer–head units. Right: selected depths in a separate adaptive-depth replay of 10 prompts, 30 layers, and chunks 0, 4, and 9 (900 records per tensor). The dashed line is the rate–distortion reference used by the stopping rule.

Depth stability. The adaptive replay selects mean depths of 10.05 for K and 7.28 for V. At a fixed layer and chunk, the across-prompt standard deviation averages 0.40 and 0.23 stages, and the mean range is 1.01 and 0.58 stages, respectively. No range exceeds two stages. This supports adapting depth to residual structure without establishing that the threshold is optimal for every backbone.

Calibration diversity. Figure 10 gives the existing calibration-size sweep. Table 8 separately increases source prompts from 20 to 100 at a fixed budget of 50,000 training vectors per layer and head. Evaluation uses the same 10 held-out prompts and frozen time-adaptive $Q = 1$ setting. Base MSE falls by 1.72% for K and 1.19% for V, without better mean video scores. Paired video p-values are 0.208/0.296/0.172 for PSNR/SSIM/LPIPS; the differences are not statistically significant. This supports the practical 20-video default, not its optimality.

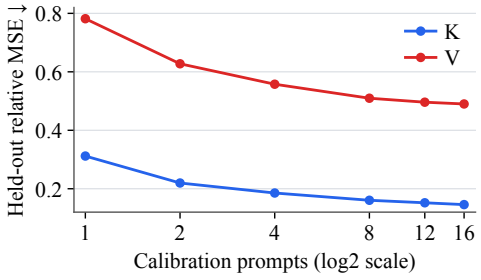


Figure 10: **Calibration-size sensitivity.** Held-out relative reconstruction error with the evaluation setting fixed.

Table 8: **Calibration diversity.** Fixed training-vector count; video scores use 10 held-out BF16 references.

Metric	20	100
K base MSE ↓	0.24440	0.24020
V base MSE ↓	0.52106	0.51484
PSNR ↑	22.8247	22.5428
SSIM ↑	0.72950	0.72293
LPIPS ↓	0.23105	0.23692

A.6 Residual Structure and Key Conditioning

K/V dispersion across prompts. We compare 1,200 matched residual records (10 prompts, 30 layers, four spans) from a fixed eight-stage, 64-prototype RVQ base with INT4 codebooks. K has higher channel-energy CV than V in every record; the median paired ratio is 10.96. This measures unevenness of channel-wise mean-square energy, not an absolute K/V energy ratio.

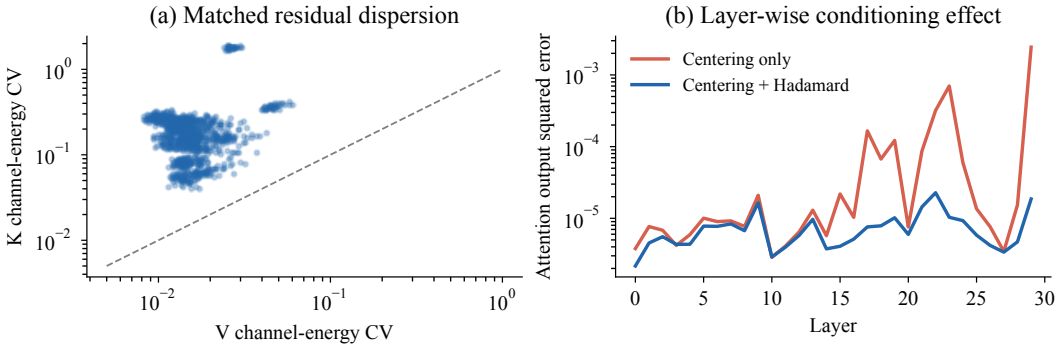


Figure 11: **Residual dispersion and attention-output sensitivity.** Left: matched raw-residual CV; the diagonal denotes equality. Right: attention-output squared error, averaged over 10 prompts and 24 heads per layer (7,200 units, standard RoPE). The panels use different statistical units.

Layer-wise effect of Hadamard mixing. Holding spatial centering fixed, Hadamard mixing reduces mean output error by 94.50%, with lower means in 28 of 30 layers. Layer 29 supplies 58.6% of the baseline error; excluding it still gives an 87.80% reduction. The effect is heterogeneous but extends beyond the last layer. These attention-replay errors do not measure independent video-quality gains.

What spatial centering removes. For a frame with n_f spatial tokens and residual mean $\boldsymbol{\mu}_f$, we report the shared-mean energy fraction $n_f\|\boldsymbol{\mu}_f\|_2^2 / \sum_j \|\mathbf{r}_{f,j}\|_2^2$. In the corresponding fixed-base diagnostic, the mean fraction is 7.63% for K and 2.79% for V. Figure 12 shows its layer dependence. This shared component motivates separating the key mean before mixing, without implying an independent LPIPS gain.

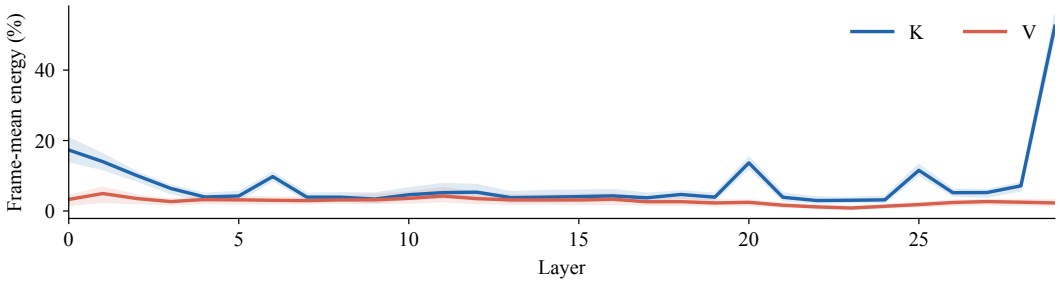


Figure 12: **Frame-mean energy in base residuals.** Layer means over 10 held-out prompts, with 10th–90th percentile bands across prompt means. V is a structural comparison; the proposed value path does not subtract its mean.

A.7 History Age and Attention Importance

We replay attention over 10 prompts, 30 layers, and nine history chunks to examine the weights in Section 3.3. Each intervention compresses one K or V chunk; all other cache tensors stay BF16. This diagnostic uses adaptive RVQ and frozen refinements.

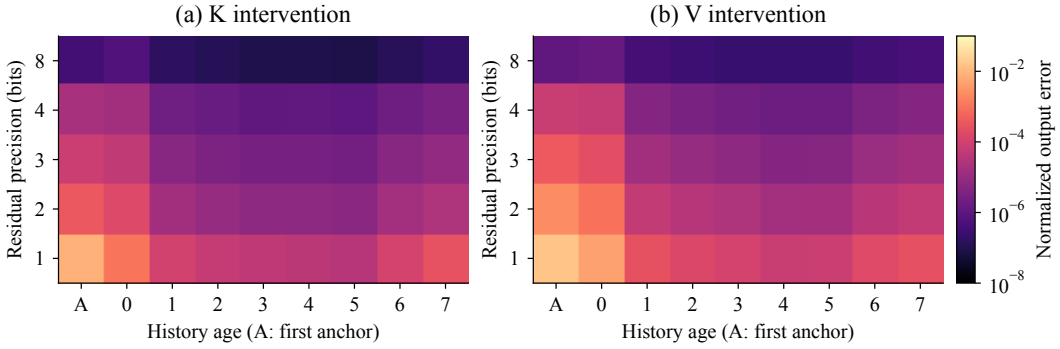


Figure 13: **Precision and temporal importance in isolated interventions.** Mean normalized output error under standard RoPE. Age increases from recent to older chunks; A is the first anchor. Both panels use the same log scale; the zero-error BF16 control is omitted.

Across ordinary ages, mean Spearman correlations of attention mass with K KL/K output error/V output error are 0.941/0.877/0.892 for standard RoPE and 0.928/0.843/0.898 for camera PRoPE, averaged over prompt–layer units and five positive precisions (anchor excluded). This supports importance-weighted distortion. The replay has no $b = 0$ intervention and does not establish multi-chunk schedule optimality.

A.8 RVQ Depth and Computational Cost

The sweep in Figure 14 uses 720 layer–head units from 10 prompts, sampling 1,024 vectors per source prompt/layer/head. Reconstruction error is mean MSE at depth L divided by mean MSE at depth zero.

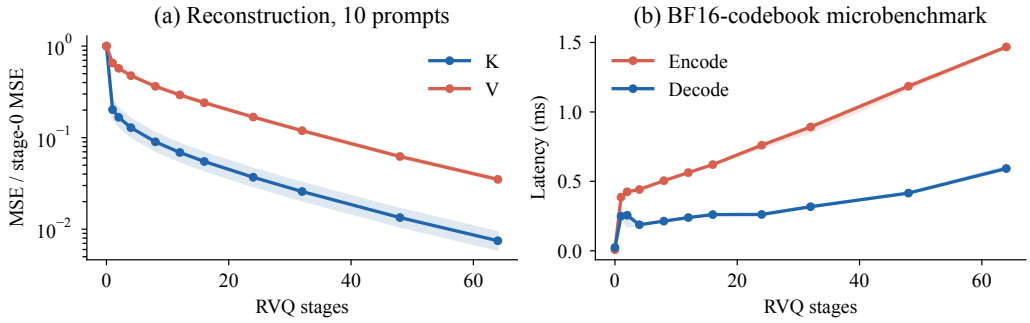


Figure 14: **Reconstruction and depth-scaling cost.** Left: K/V normalized MSE with 95% paired-bootstrap intervals (720 units, 10,000 resamples). Right: median encode/decode latency with interquartile bands; BF16-codebook microbenchmark.

The timing input is one K tensor of shape $[1, 24, 3520, 128]$ from prompt 0, layer 0, on RTX 6000 Ada: 10 warm-up and 100 timed iterations, CUDA synchronization around each region, no file/codebook loading. The timed RVQ operations use BF16 codebooks and scales. These are depth-cost diagnostics, not separate K/V timings or production end-to-end measurements.

A.9 Additional Fixed-Frame Video Comparisons

We take prompt IDs 00–03 in order from the archived 10-prompt gallery, excluding main-text prompt 05. All methods use seed 42, 50 steps, 173 frames at 1280×704, and the same fixed pairs $45 \rightarrow 46$ and $109 \rightarrow 110$ (zero-based). Heatmaps share the $[0, 0.10]$ scale; invalid pixels are white. Curves show all 172 pairs under Section A.3's TWE definition. Cases where SuBitKV exceeds BF16 error are retained. $Q = 0.2$ specifies residual refinement, not total cache storage; these examples do not replace the aggregate scores.

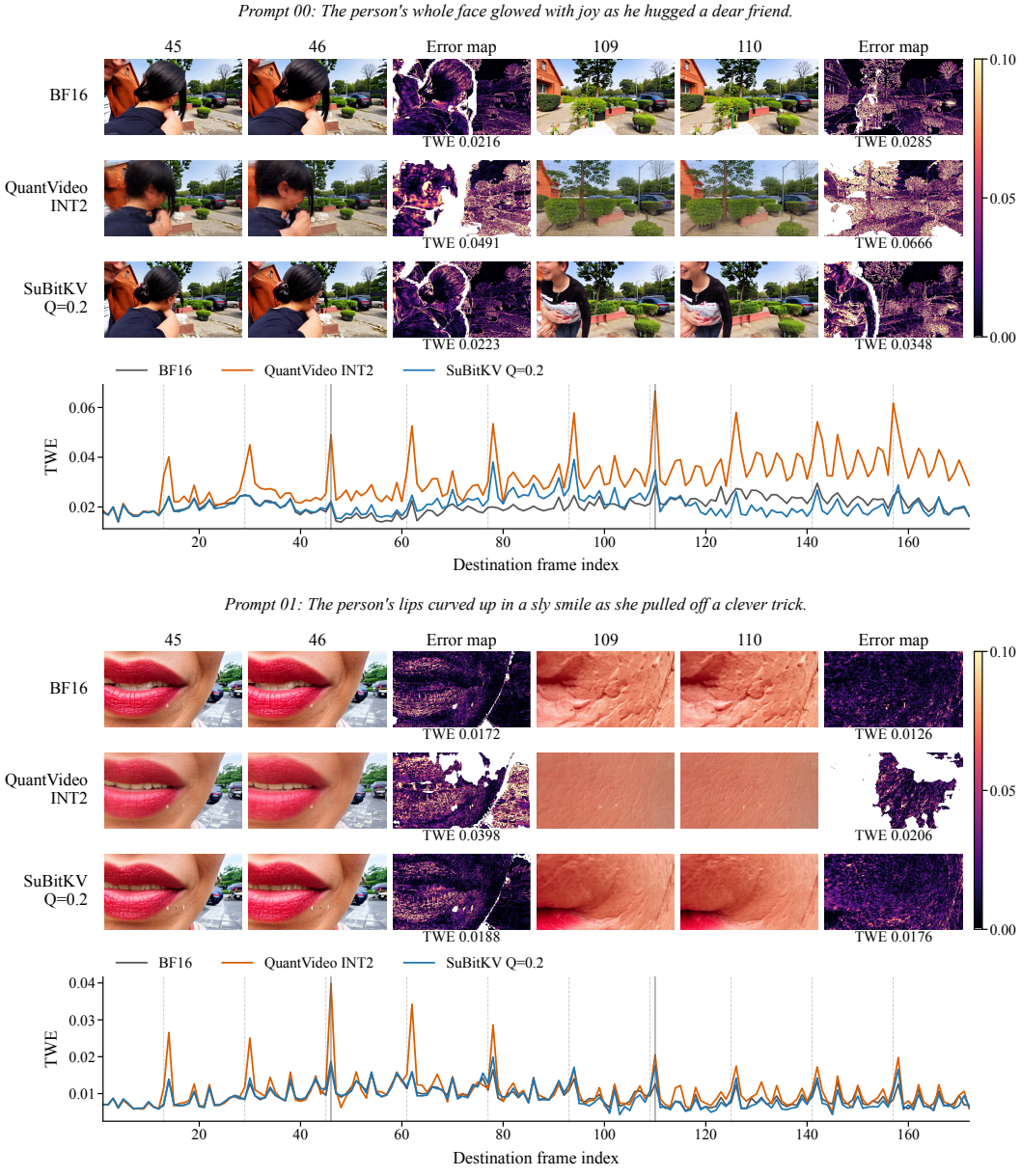


Figure 15: **Fixed-frame comparisons, prompts 00 and 01.** Rows show BF16, QuantVideo INT2, and SuBitKV $Q = 0.2$. Each group contains the two frames and the motion-aligned error map. TWE curves below each example use the same archived optical-flow evaluation; dashed lines mark chunk boundaries and solid lines mark the displayed destination frames.

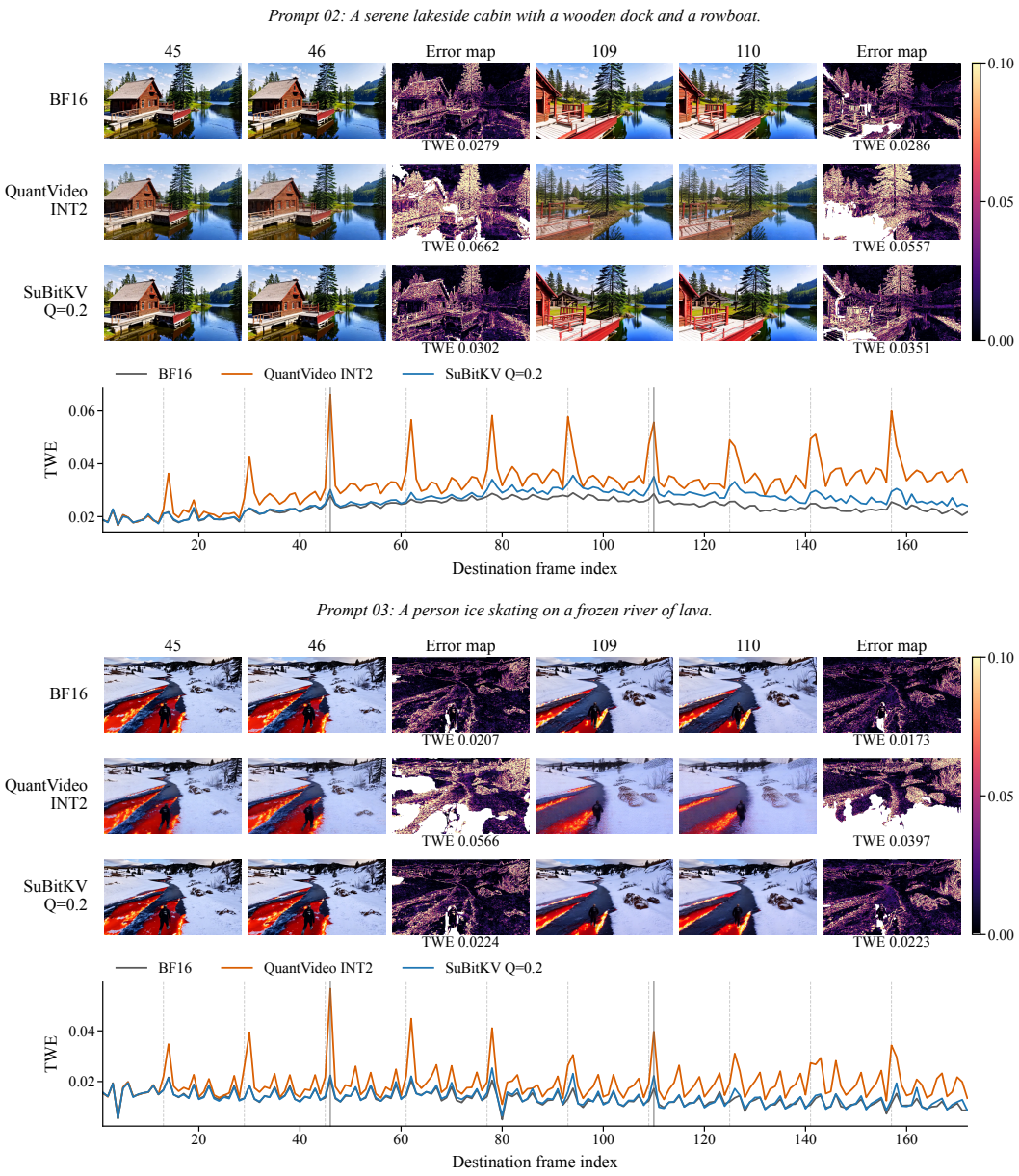


Figure 16: **Fixed-frame comparisons, prompts 02 and 03.** The selection rule, method order, frame indices, and heatmap scale are identical to Figure 15. Both ordinary motion and local appearance changes are retained. The full curves provide context for the two predetermined frame pairs.