MEMORY-EFFICIENT AUTOREGRESSIVE VIDEO GENERATION VIA HIERARCHICAL MIXED-PRECISION KV CACHE QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive video generation models face a critical KV cache memory bottleneck as the demand for long video synthesis grows. Existing LLM quantization methods perform poorly when applied to video generation models due to the high dynamic range of video KV cache activations and their heightened sensitivity to quantization errors. We propose HiMix, a hierarchical mixed-precision KV cache quantization framework for memory-efficient autoregressive video generation. Specifically, HiMix first determines layer-wise bit budgets offline based on per-layer sparsity analysis, then performs runtime channel-aware quantization guided by Q-K saliency: important channels receive higher bit-widths (4-bit) while less important ones are quantized to 2-bit or 1-bit, with spatiotemporal Hadamard smoothing applied prior to quantization to reduce error. Experiments on the Self-Forcing and Causal Forcing models demonstrate that HiMix achieves nearly 7.1$\times$ compression ratio for KV cache while preserving video generation quality and long-range consistency comparable to the BF16 baseline, substantially reducing memory consumption.

1 INTRODUCTION

Diffusion transformers (DiTs) (Peebles & Xie, 2023) underpin advances in high-quality video generation, including CogVideoX (Yang et al., 2025), HunyuanVideo (Kong et al., 2024), and Wan (Team Wan et al., 2025). While these models synthesize coherent video clips, bidirectional denoising processes the full sequence jointly and is poorly suited to streaming output. Autoregressive video diffusion models (Chen et al., 2024; Huang et al., 2025; Yin et al., 2025; Zhu et al., 2026) instead generate frames or chunks causally, reusing historical keys and values through a KV cache. This reuse avoids repeatedly processing past context and supports interactive video generation (Feng et al., 2026; Shin et al., 2026) and long-horizon world modeling (Sun et al., 2026; Mao et al., 2025).

However, this computational reuse creates a persistent KV cache memory bottleneck. Unlike model weights, whose storage is fixed, the cache grows with the retained generation history. Moreover, higher spatial resolutions further inflate the number of tokens per frame. Quant VideoGen (QVG) (Xi et al., 2026) reports roughly 34 GB of BF16 KV cache for 38K latent tokens in a 5-second 480p LongCat-Video sequence (Meituan LongCat Team et al., 2025). Truncating history bounds memory but removes context that can support long-horizon consistency (Hong et al., 2025). Reducing the precision of stored keys and values offers a different trade-off: it retains historical positions while reducing their representation cost.

Low-bit KV quantization is well established in LLMs (Liu et al., 2024; Hooper et al., 2024; Kang et al., 2024), with rotation-based approaches redistributing outliers before quantization (Ashkboos et al., 2024; Su et al., 2025). However, KV cache outlier distributions differ substantially between LLMs and video models. Directly applying LLM quantization methods to video models therefore yields poor performance (Xi et al., 2026). QVG (Xi et al., 2026) addresses this variation through semantic-aware smoothing and progressive residual quantization at 2 bits. Nevertheless, QVG applies a uniform quantization strategy across all DiT layers and channels, overlooking differences in their quantization sensitivity.

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.

In this work, we propose **HiMix**, a training-free, hierarchical mixed-precision KV cache quantization framework for autoregressive video generation. HiMix combines offline layer-wise budget allocation with runtime channel-wise precision assignment, using spatiotemporal Hadamard smoothing to redistribute token-dependent outliers before quantization.

Specifically, we first provide a sparsity-guided offline budget allocation module to account for differences in quantization sensitivity across DiT layers. Luo et al. (2026) identify input-stable, layer-dependent attention sparsity. Our calibration analysis further links attention sparsity to attention-output error, supporting sparsity as an empirical sensitivity proxy. Based on this observation, we assign different precision budgets to sensitive, normal, and robust layers. These budgets determine the channel counts at each precision and remain fixed during generation.

Moreover, we introduce query-aware channel ranking to determine which channels receive each precision within the offline layer budgets. Channel importance depends on both keys and the current queries. A channel with a modest key range can still contribute strongly to attention scores when its query magnitude is large. Key-only ranking can therefore miss important channels. We use joint Q-K importance scores to rank key cache channels and assign them to 4-, 2-, or 1-bit groups at runtime. The Value cache uses uniform per-token quantization. Together, the two modules adapt precision across layers and channels without fine-tuning model weights or discarding historical tokens.

Our main contributions are as follows:

- We introduce HiMix, a training-free, hierarchical mixed-precision KV cache quantization framework for memory-efficient autoregressive video generation. It allocates precision across layers and channels while retaining the complete generation history.
- We use attention sparsity to assign offline layer-wise precision budgets and joint Q-K importance scores to rank channels for runtime mixed-precision quantization. Spatiotemporal Hadamard smoothing reduces KV cache outliers before quantization.
- Experiments on Self-Forcing and Causal Forcing demonstrate nearly 7.1$\times$ KV cache compression at approximately 2 bits on average, while preserving generation quality comparable to the BF16 baseline with less than 4% additional end-to-end latency.

2 Related Work

2.1 KV Cache Quantization for LLMs

KV cache quantization reduces memory consumption by representing cached keys and values at lower precision. KIVI (Liu et al., 2024) exploits different key and value distributions through per-channel key and per-token value quantization. KVQuant (Hooper et al., 2024) combines non-uniform quantization with high-precision outlier retention, whereas GEAR (Kang et al., 2024) compensates errors with low-rank and sparse corrections. QuaRot (Ashkboos et al., 2024) and RotateKV (Su et al., 2025) use rotations to mitigate outliers and improve low-bit KV cache quantization.

Mixed-precision methods allocate precision at different granularities within the KV cache. MiKV (Yang et al., 2024) retains important tokens at higher precision and stores less important tokens at lower precision rather than discarding them. AsymKV (Tao et al., 2024) accounts for the different effects of key and value quantization on attention outputs through layer-wise asymmetric precision configurations. ChanMix (Liao & Wen, 2026) explores channel-aware allocation and retrieval heads. MixKVQ (Zhang et al., 2025) combines query magnitudes with key quantization difficulty to identify important channels. However, these methods primarily target the cache distributions and inference patterns of LLMs. Their applicability to autoregressive video generation requires further validation under token-dependent outliers and spatiotemporal attention patterns.

2.2 Quantization for Video Generation

Quantization reduces the memory and computation costs of video diffusion models. Early diffusion PTQ methods, including PTQ4DM (Shang et al., 2023) and Q-Diffusion (Li et al., 2023), provide a foundation for low-precision diffusion inference. For image and video DiTs, ViDiT-Q (Zhao et al., 2025) combines fine-grained quantization with layer- and timestep-sensitive mixed precision, while

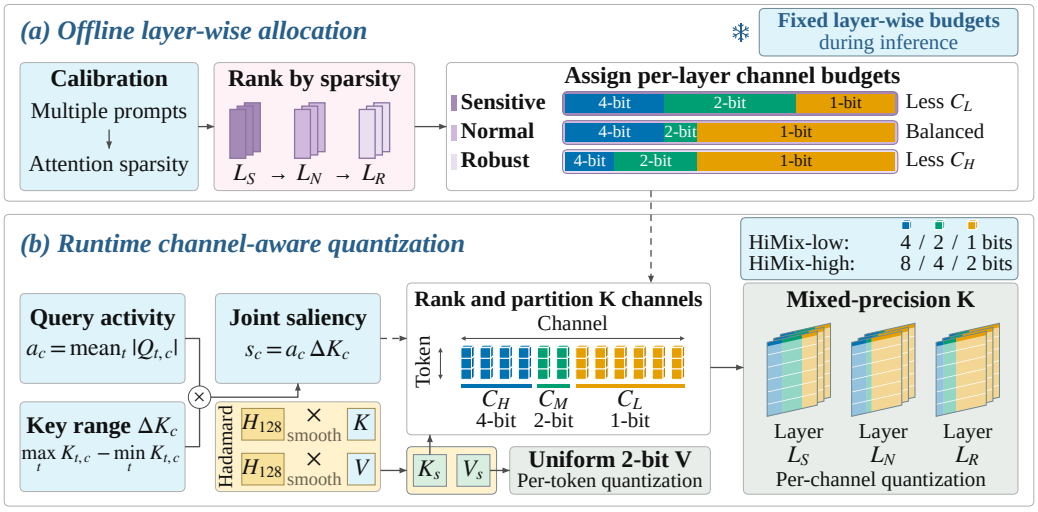


Figure 1: Overview of HiMix. (a) Offline calibration assigns per-layer channel budgets according to attention sparsity. (b) Runtime saliency guides channel ranking and mixed-precision key quantization. Hadamard smoothing is applied separately to key and value before quantization.

Q-DiT (Chen et al., 2025) adapts quantization granularity and activation parameters to channel and temporal variation. Related diffusion quantization work, SVDQuant (Li et al., 2025), absorbs outliers into a high-precision low-rank branch. DVD-Quant (Li et al., 2026) targets video DiTs with data-free PTQ, combining grid refinement, rotated quantization, and adaptive bit switching. QVGen (Huang et al., 2026) instead uses quantization-aware training, with auxiliary modules to improve optimization and rank decay to remove their inference overhead. For autoregressive video generation, Q-ARVD (Tang et al., 2026) weights calibration errors by frame-wise sensitivity and handles outliers through adaptive dual-scale quantization. However, weight and activation quantization does not directly address the growing historical KV cache in autoregressive inference.

For KV cache quantization in autoregressive video generation, Quant VideoGen (Xi et al., 2026) directly quantizes the historical KV cache using semantic-aware smoothing and progressive residual quantization at a common 2-bit target. Tuncer et al. (2026) identify quantization-induced attention bias and correct it using key quantization step sizes and query norms. However, these methods apply a uniform precision policy across layers and channels, without analyzing inter-layer differences in KV cache quantization sensitivity.

3 Method

HiMix addresses three sources of KV cache quantization error: token-dependent outliers, differences in layer sensitivity, and variation in channel importance. We use spatiotemporal Hadamard smoothing to reduce outliers (Section 3.1) and organize precision allocation hierarchically across layers and channels. At the layer level, offline sparsity analysis determines each layer's channel budgets (Section 3.2). Within these budgets, Q-K joint saliency assigns precision to individual channels at runtime (Section 3.3). Figure 1 summarizes the offline and runtime stages.

3.1 Spatiotemporal Hadamard Smoothing

Video KV cache outliers vary along the token sequence. For example, the Llama-3.2-1B Key cache exhibits persistent channel-wise outliers, whereas the Self-Forcing-1.3B Key cache varies substantially across tokens even within the same channel (Figure 2). The Self-Forcing-1.3B Value cache is more uniform overall, but isolated outliers remain. The surfaces show absolute magnitudes over tokens and channels, with separate vertical scales. These token-dependent extremes motivate smoothing along the sequence dimension before low-bit quantization.

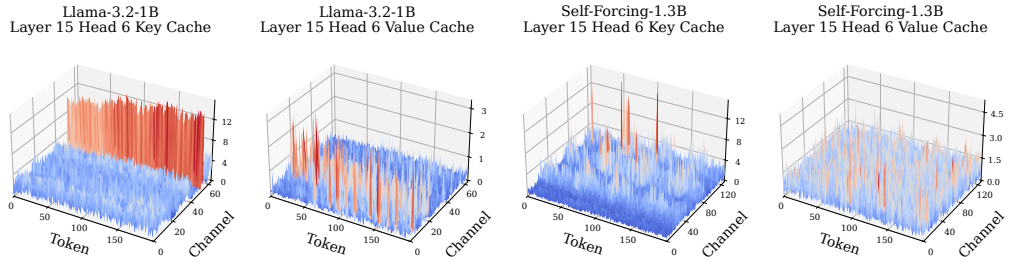


Figure 2: Key Cache and Value Cache activation magnitudes in Llama-3.2-1B and Self-Forcing-1.3B at layer 15, head 6.

To redistribute the outliers within each channel, we apply a spatiotemporal Hadamard transform independently to key and value. Given Key and Value cache chunks $K, V \in \mathbb{R}^{B\times S\times H\times d}$, we flatten the head and head-dimension axes into a channel axis of size $C = Hd$, obtaining $K, V \in \mathbb{R}^{B\times S\times C}$. For each channel c, we partition the token sequence into consecutive blocks of $h = 128$ tokens and apply a normalized Hadamard matrix independently to each block for smoothing:

$$(K_s)_{b,i} = W_h K_{b,i}, \qquad (V_s)_{b,i} = W_h V_{b,i}, \qquad W_h = \frac{1}{\sqrt{h}} H_h, \tag{1}$$

where $K_{b,i}, V_{b,i} \in \mathbb{R}^{h\times C}$ denote the i-th token blocks of the b-th batch element, with each column corresponding to one channel, and $H_h \in \{\pm 1\}^{h\times h}$ is the Sylvester Hadamard matrix. Concatenating the smoothed blocks yields the tensors K_s and V_s. This block-wise sequence-domain transform smooths outlier magnitudes across contiguous serialized video tokens, producing a more quantization-friendly distribution.

Because $W_h^\top W_h = I$ and $W_h = W_h^\top$, the normalized transform is self-inverse: $W_h^{-1} = W_h$. Accordingly, after quantization and dequantization, each complete block is recovered as

$$\widehat{K}_{b,i} = W_h\, Q^{-1}(Q((K_s)_{b,i}))\,, \qquad \widehat{V}_{b,i} = W_h\, Q^{-1}(Q((V_s)_{b,i}))\,. \tag{2}$$

The remaining reconstruction error therefore arises from quantization rather than the normalized transform itself.

3.2 Offline Layer-Wise Precision Budget Allocation

Smoothing reduces activation outliers, but DiT layers still differ in quantization sensitivity. SVOO (Luo et al., 2026) defines attention density as the minimum fraction of attention entries needed to cover 80% of the cumulative attention mass. Its analysis reveals stable layer-wise density patterns across prompts and pronounced heterogeneity between layers. We assign layer-wise precision budgets offline using attention sparsity, $S = 1 - D$, as a proxy for quantization error, where D denotes attention density.

To measure sensitivity, we compare the attention output before and after quantizing the historical KV cache. For layer l, the relative quantization error is

$$\varepsilon_l = \frac{\|\widehat{O}_l - O_l\|_F}{\|O_l\|_F}, \tag{3}$$

where O_l and $\widehat{O}_l$ are the attention outputs before and after quantization, respectively.

To investigate differences in KV cache sensitivity across layers, we analyze key-only and value-only quantization separately, as shown in Figure 3a. We quantize either key or value in a single layer and evaluate the resulting attention-output relative error in (3). These errors define layer rankings for each quantization scheme. Meanwhile, we also incorporate the relationship between attention sparsity and layer ranking into our analysis. In Figure 3a, each point represents the Spearman correlation coefficient between layer rankings under two different prompts. Among the three measurements, sparsity and key-only quantization error remain relatively stable across prompts. Figure 3b shows that sparsity and the key-only quantization error exhibit a positive linear correlation. Since the

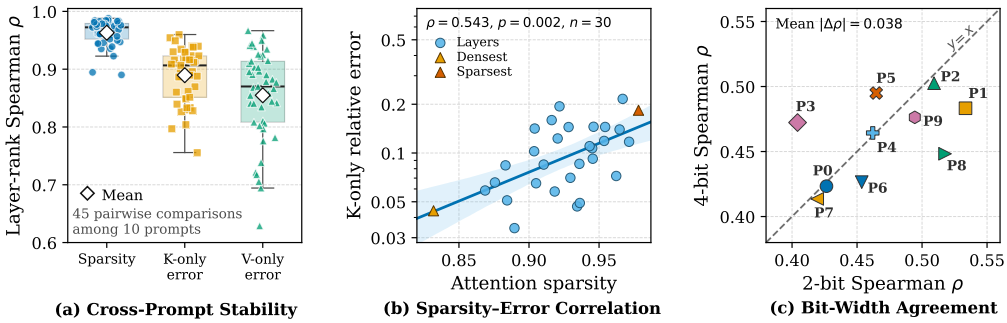


Figure 3: (a) Cross-10 prompts stability of layer rankings by attention sparsity and quantization error. (b) Relationship between attention sparsity and key-only attention-output relative error. (c) Stability of the relationship between sparsity and quantization error across quantization precisions.

measurement of sparsity does not require quantizing and reconstructing the cache, or recomputing attention outputs for different quantization methods and precision, we therefore choose attention sparsity as a proxy for the measurement of key quantization sensitivity.

We rank layers by attention sparsity and apply K-means clustering to their sparsity scores to obtain three groups of layers (Figure 1a). The cluster with the highest mean sparsity contains the sensitive layers L_S. The cluster with the lowest mean sparsity contains the robust layers L_R. The remaining cluster contains the normal layers L_N. Figure 3c shows that the correlation between sparsity and quantization error remains stable across different quantization bit-widths. This inspires us to adopt mixed precision to allocate budgets across layers. We design two configurations, HiMix-low and HiMix-high, approximately corresponding to the compression ratios of 2-bit uniform quantization and 4-bit uniform quantization, respectively. Let C_H, C_M, and C_L be the numbers of high-, medium-, and low-precision key channels. In HiMix-low, C_H, C_M, and C_L correspond to the number of 4-, 2-, and 1-bit channels. In HiMix-high, they correspond to the number of 8-, 4-, and 2-bit channels. For normal layers L_N with total number of channels C^N_{total}, we set the specific C^N_H, C^N_M and C^N_L as

$$(C^N_H, C^N_M, C^N_L) = C^N_{\text{total}} \cdot \left(\frac{\alpha}{2},\, 1 - \frac{3\alpha}{2},\, \alpha\right), \tag{4}$$

where $C^N_{\text{total}} = C^N_H + C^N_M + C^N_L$ and α is the fraction of low-precision channels. We choose $\alpha = 0.6$ based on ablation experiments. For sensitive layers L_S with total number of channels C^S_{total}, we increase the layer precision by setting $C^S_L = C^N_L/2$, yielding $(C^S_H, C^S_M, C^S_L) = C^S_{\text{total}} \cdot (\alpha/2,\, 1 - \alpha,\, \alpha/2)$. For robust layers L_R with total number of channels C^R_{total}, we reduce the precision budget by setting $C^R_H = C^N_H/2$, yielding $(C^R_H, C^R_M, C^R_L) = C^R_{\text{total}} \cdot (\alpha/4,\, 1 - 5\alpha/4,\, \alpha)$. The resulting channel counts are fixed offline.

In contrast to attention sparsity and the key-only quantization error, the value-only quantization error varies noticeably across prompts (Figure 3a). In addition, the magnitudes of the Value cache are relatively uniform within each layer and remain small in absolute value (Figure 2). Therefore, we adopt uniform quantization for Value caches.

3.3 Runtime Channel-Aware Mixed-Precision Quantization

Offline budget allocation determines the number of channels at each precision within each layer. The precision of each channel within a layer must then be determined through channel ranking. Quantization error is usually directly affected by the quantization scale, so ChanMix (Liao & Wen, 2026) ranks channels by the key range $\Delta K_c = \max_t K_{t,c} - \min_t K_{t,c}$, where t indexes tokens. However, video quality is directly related to attention outputs, and attention depends on both query and key. A key channel with a large magnitude range but small corresponding query magnitudes has little influence on attention (Figure 4a) and is therefore not sensitive. Ranking channels solely by ΔK_c is thus insufficient.

To quantify this discrepancy, we compare the top 10% of channels selected by ΔK_c with those selected by the query-conditioned score $|Q_{t,c}|\Delta K_c$ for the same head and query token. Denoting

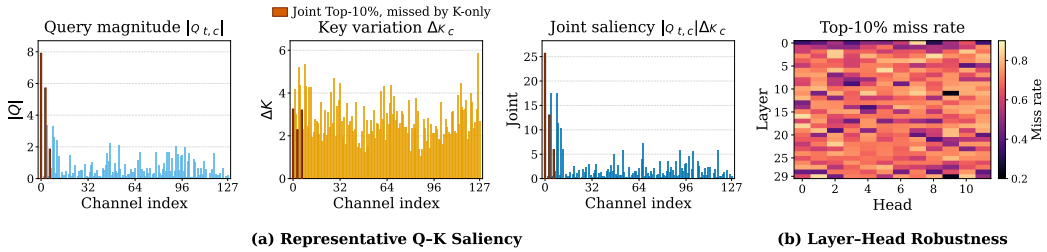


Figure 4: Q-K channel saliency in Self-Forcing-1.3B. (a) Query magnitude, key range, and their product for a representative token. Orange bars highlight selected channels missed by key-only ranking. (b) Mean top-10% miss rate relative to the joint ranking for each layer–head pair, averaged over sampled tokens and blocks from one inference trajectory.

these sets by $\mathcal{T}_K$ and $\mathcal{T}_{QK}$, respectively. We define $\mathrm{MissRate}_{10\%} = 1 - |\mathcal{T}_K \cap \mathcal{T}_{QK}|/|\mathcal{T}_{QK}|$. This is the fraction of jointly salient channels omitted by key-only selection. As shown in the heatmap in Figure 4b, such omissions occur in almost all layers and heads. This motivates us to rank channels using joint query and key importance scores.

For runtime allocation, we average query magnitudes across the most recently captured token block to obtain a shared channel ranking. For a channel c spanning S tokens, its joint query and key importance score is

$$s_c = \left(\frac{1}{S} \sum_{t=1}^{S} |Q_{t,c}| \right) \Delta K_c, \tag{5}$$

For a given layer's Key cache, we rank its C_{total} channels by s_c and partition them into three groups of C_H, C_M, and C_L channels according to the offline budget. The three groups undergo per-channel asymmetric quantization at different precisions after smoothing. For the Value cache, we use uniform per-token quantization at 2 bits in the low-bit configuration and 4 bits in the high-bit configuration. Thus, layer budgets remain fixed while channel assignments adapt to the current Q-K activations. Quantization is triggered whenever the KV cache accumulates a specified number of new chunks. We apply Hadamard smoothing to both key and value. Using query and key captured while storing the latest chunk, we compute importance scores for Key cache channels and rank them, followed by channel reordering, quantization, and packing. Quantized data remain in the KV cache, while newly stored chunks are kept in BF16.

3.4 Efficient Implementation

We fuse cache unpacking, dequantization, and inverse key channel reordering in a Triton kernel. Packing shifts quantized integer codes into non-overlapping bit fields within uint8 storage, using precomputed byte indices and bit offsets. During reconstruction, right shifts and bit masks extract each code, followed by dequantization with per-group key or per-token value scales and zero-points. The inverse channel permutation maps each original key channel to its stored code, allowing direct writes in the original channel order without materializing full-sized unpacked integer tensors or a separate reordering buffer. Forward smoothing uses a normalized butterfly Hadamard transform, while the inverse remains a separate batched matrix multiplication. The fused kernel preserves intermediate dtype conversions and rounding in the reconstruction path.

4 Experiments

4.1 Experimental Setup

Models. We evaluate HiMix on Self-Forcing (Huang et al., 2025) and Causal Forcing (Zhu et al., 2026). We use a chunk size of 3 frames and condition each chunk on all previously generated frames.

Metrics. We use VBench (Huang et al., 2024) to report Background Consistency, Image Quality, Subject Consistency, and Aesthetic Quality. Throughout all subsequent tables, the average score

Table 1: Autoregressive video-generation quality and KV cache compression benchmark. VBench quality scores range from 0 to 100. Higher values are better. Average scores are computed over the four reported VBench dimensions.

Method	Compression Ratio (BF16)	Background Consistency	Image Quality	Subject Consistency	Aesthetic Quality	Average Score
Self-Forcing: 2-bit KV Cache						
BF16	1.00×	93.92	69.35	92.13	62.57	79.49
KIVI (Liu et al., 2024)	6.40×	92.89	68.00	90.23	60.69	77.95
QuaRot (Ashkboos et al., 2024)	6.40×	92.89	67.03	89.92	60.66	77.63
QVG (Xi et al., 2026)	6.96×	93.73	69.19	91.77	62.25	79.24
HiMix-low	**7.09**×	93.63	69.55	91.70	62.39	**79.32**
Self-Forcing: 4-bit KV Cache						
BF16	1.00×	93.92	69.35	92.13	62.57	79.49
KIVI (Liu et al., 2024)	3.55×	93.58	68.72	91.47	62.12	78.97
QuaRot (Ashkboos et al., 2024)	3.55×	93.88	69.31	92.07	62.56	79.46
QVG (Xi et al., 2026)	3.72×	93.91	69.32	92.13	62.55	79.48
HiMix-high	**3.80**×	93.89	69.33	92.11	62.57	**79.48**
Causal-Forcing: 2-bit KV Cache						
BF16	1.00×	93.00	69.94	90.92	62.87	79.18
KIVI (Liu et al., 2024)	6.40×	91.59	67.82	87.94	59.98	76.83
QuaRot (Ashkboos et al., 2024)	6.40×	91.64	68.01	87.89	60.13	76.92
QVG (Xi et al., 2026)	6.96×	92.88	69.41	90.59	62.63	78.88
HiMix-low	**7.09**×	92.78	69.96	90.56	62.55	**78.96**
Causal-Forcing: 4-bit KV Cache						
BF16	1.00×	93.00	69.94	90.92	62.87	79.18
KIVI (Liu et al., 2024)	3.55×	92.66	68.97	90.22	62.23	78.52
QuaRot (Ashkboos et al., 2024)	3.55×	93.01	69.87	90.95	62.86	79.17
QVG (Xi et al., 2026)	3.72×	92.99	69.89	90.92	62.91	79.18
HiMix-high	**3.80**×	92.99	69.98	90.95	62.95	**79.22**

refers to the mean of these four VBench dimensions. We additionally report the KV cache compression ratio and the end-to-end generation overhead.

Datasets. We generate videos using 1,003 prompts provided by MovieGen (Polyak et al., 2024). Specifically, we adopt the extended MovieGen prompt set released with the official Self-Forcing implementation for model evaluation.

Compared methods. For both models, we compare BF16, KIVI (Liu et al., 2024), QuaRot (Ashkboos et al., 2024), Quant VideoGen (QVG) (Xi et al., 2026), and HiMix. All quantization methods use group size 64 for a fair comparison.

Implementation Details. Experiments use a memory-modified RTX 4090 with 48 GB of VRAM. Following QVG, all models retain the full KV cache throughout generation. Because quantization substantially reduces cache memory, HiMix can preserve the complete generation history without truncating earlier context, improving visual quality and mitigating long-horizon drift while retaining the memory savings of low-bit KV cache storage. We use Pre-RoPE quantization: all quantization and dequantization operations occur before rotary position embedding, and key per-group scaling factors are stored in FP16.

4.2 Experimental Results

Quality evaluation. Table 1 demonstrates that HiMix consistently achieves a favorable quality–memory trade-off across both Self-Forcing and Causal Forcing. Under the 2-bit setting, HiMix-low delivers the strongest overall quality among the low-bit baselines while attaining a 7.1× KV cache compression ratio. Its improvements are consistent across the VBench dimensions, indicating that the proposed hierarchical allocation preserves both visual fidelity and temporal consistency rather than optimizing a single metric. Under the higher-precision setting, HiMix-high achieves quality comparable to the BF16 baseline and remains competitive with the strongest quantization baselines



Figure 5: Qualitative comparison of 30-second Self-Forcing videos generated with BF16, KIVI, QVG, and HiMix.

while retaining a 3.8× compression ratio. These results verify that HiMix generalizes across autoregressive generation frameworks and provides scalable operating points for different memory budgets. Together, these comparisons support the advantages and generality of the hierarchical allocation framework across the evaluated models and precision settings.

Long-video evaluation. We further evaluate all methods under an extended generation horizon using Self-Forcing. Each method generates 30-second videos from the same evaluation prompts, and we measure Image Quality and Subject Consistency with VBench. As reported in Table 2, HiMix achieves the best Image Quality while preserving Subject Consistency comparable to the BF16 baseline and QVG. In contrast, directly applying KIVI or QuaRot causes substantial degradation in both metrics. These results demonstrate that HiMix maintains image fidelity and long-range subject consistency over extended autoregressive generation horizons.

Table 2: Long-video evaluation on Self-Forcing.

Method	Image Quality ↑	Subject Consistency ↑
BF16	67.81	**89.99**
KIVI	62.32	84.22
QuaRot	61.86	85.30
QVG	67.23	89.34
HiMix	**68.05**	89.24

Figure 5 qualitatively compares the stability of image quality and subject consistency among full-precision and quantized methods during long-video generation. The forest and coast examples appear on the left and right, respectively, with rows ordered as BF16, KIVI, QVG, and HiMix. Dashed boxes mark regions with visual degradation or changes in scene content in the two examples. As the video length increases, KIVI produces increasingly blurred frames, resulting in a rapid decline in image quality. QVG preserves visual quality initially, but the subject disappears in the later portion of the video, breaking long-range subject consistency. In contrast, HiMix maintains stable visual content and long-range subject consistency despite its high KV cache compression ratio.

4.3 Ablation Studies

Component ablation. We conduct a cumulative ablation study on Self-Forcing to assess the contributions of Hadamard smoothing, channel-aware allocation, and layer-wise heterogeneous allocation. Starting from HiMix, each subsequent configuration removes one additional component. The final configuration applies simple per-channel quantization to key and per-token quantization to value.

Table 3: Cumulative ablation on Self-Forcing.

Configuration	Had.	Chan.	Layer	Avg.	Comp.
HiMix	✓	✓	✓	**79.32**	**7.09**×
– Layer alloc.	✓	✓	–	79.22	6.96×
– Channel alloc.	✓	–	–	78.83	6.40×
– Hadamard	–	–	–	77.95	6.40×

As summarized in Table 3, progressively removing layer-wise allocation, channel-aware allocation, and Hadamard smoothing lowers generation quality at each stage. Removing the allocation components also reduces the compression ratio, indicating that layer-wise and channel-aware allocation help preserve generation quality while reducing the storage cost of the KV cache. The complete HiMix configuration achieves the strongest overall quality and highest compression ratio among these configurations.

Quantization group size. The quantization group size controls how many tokens share quantization parameters, trading metadata overhead against reconstruction fidelity. Table 4 compares group sizes of 32, 64, and 128. Smaller groups incur more metadata overhead, whereas larger groups improve compression at some cost to video quality. Among the tested settings, a group size of 64 achieves the highest average quality while retaining substantial compression. We therefore use 64 by default to balance generation quality and KV cache memory savings.

Table 4: Effect of quantization group size on video quality and KV cache compression.

Group Size	Average Score	Compression Ratio
32	79.19	6.46×
64	**79.32**	**7.09×**
128	79.17	7.49×

Sensitivity to α. We evaluate the sensitivity of HiMix-low to the low-precision channel fraction α on Self-Forcing (Table 5). Image Quality improves from $\alpha = 0.2$ to 0.6, then decreases at 0.66, near the feasible upper limit of $2/3$. Increasing the low-precision channel fraction also increases the number of high-precision channels, which may improve video quality. However, an excessive proportion of low-precision channels can still degrade the final video quality, making a balanced allocation necessary. We therefore choose $\alpha = 0.6$ as the default low-precision channel fraction.

Table 5: Sensitivity to α on Self-Forcing.

α	0.20	0.40	**0.60**	0.66
Image Quality	69.13	69.41	**69.55**	69.39

4.4 Efficiency Evaluation

KV cache memory. Under BF16 precision, the KV cache grows by approximately 6.6 GB for every eight generated chunks. Consequently, generating a 60-second video requires about 65 GB of KV cache memory, which exceeds the capacity of most commodity GPUs. HiMix reduces this memory footprint to 9.3 GB while retaining the full historical context, thereby preserving long-range consistency throughout the generated video. The reported memory usage includes the quantized data, scales, zero-points, and channel-reordering indices.

End-to-end latency. We evaluate the end-to-end latency of HiMix using Self-Forcing on two GPU architectures to assess its hardware robustness. On each GPU, latency is measured by generating a video of 63 latent frames. As shown in Table 6, the baseline end-to-end generation takes 52 s on an RTX 4090, while HiMix adds only 1.89 s, corresponding to a 3.6% overhead. On an RTX 3090, the generation takes 86 s and HiMix incurs an additional 3.22 s, corresponding to a 3.7% overhead. The consistently low overhead across both GPUs demonstrates that HiMix introduces negligible additional cost under different hardware settings.

Table 6: Latency overhead across GPUs. Baseline time excludes the HiMix quantization cost.

GPU	Baseline (s)	Extra (s)	Overhead (%)
RTX 4090	52	1.89	3.6
RTX 3090	86	3.22	3.7

5 Conclusion

We presented HiMix, a hierarchical mixed-precision KV cache quantization framework for autoregressive video generation. HiMix uses attention sparsity to allocate bit budgets across layers and joint Q-K saliency to assign precision to channels within each layer, while spatiotemporal Hadamard smoothing mitigates activation outliers. Experiments on Self-Forcing and Causal Forcing show that HiMix achieves 7.09× KV cache compression while maintaining near-lossless generation quality, with less than 4% end-to-end latency overhead. These results demonstrate that jointly adapting precision across layers and channels enables efficient long-horizon video generation with substantially reduced memory consumption during autoregressive inference.

Ethics Statement

The research conducted in the paper conforms, in every respect, with the ICLR Code of Ethics.

Reproducibility Statement

Implementation details and experimental settings are provided in Sections 3 and 4. The appendices describe the diagnostic measurements, budget derivation, and online KV cache processing algorithm. We will release the code to facilitate reproduction of our results.

References

Saleh Ashkboos, A. Mohtashami, M. L. Croci, B. Li, P. Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and J. Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24081–24125, 2024.

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-DiT: Accurate post-training quantization for diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28306–28315, 2025.

T. Feng, Z. Li, S. Yang, H. Xi, M. Li, X. Li, L. Zhang, K. Yang, K. Peng, S. Han, M. Agrawala, K. Keutzer, A. Kodaira, and C. Xu. Streamdiffusionv2: A streaming system for dynamic and interactive video generation. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 1656–1669, 2026.

Y. Hong, Y. Mei, C. Ge, Y. Xu, Y. Zhou, S. Bi, Y. Hold-Geoffroy, M. Roberts, M. Fisher, E. Shechtman, K. Sunkavalli, F. Liu, Z. Li, and H. Tan. RELIC: Interactive video world model with long-horizon memory, 2025.

Coleman Hooper, Sehoon Kim, H. Mohammadzadeh, Michael W. Mahoney, Y. S. 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.

X. Huang, Z. Li, G. He, M. Zhou, and E. Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, volume 38, pp. 167283–167308, 2025.

Yushi Huang, Ruihao Gong, Jing Liu, Yifu Ding, Chengtao Lv, Haotong Qin, and Jun Zhang. QVGen: Pushing the limit of quantized video generative models. In *International Conference on Learning Representations (ICLR)*, 2026.

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 (CVPR)*, pp. 21807–21818, 2024.

H. Kang, Q. Zhang, S. Kundu, G. Jeong, Z. Liu, T. Krishna, and T. Zhao. GEAR: An efficient KV cache compression recipe for near-lossless generative inference of LLM, 2024.

Weijie Kong et al. HunyuanVideo: A systematic framework for large video generative models, 2024.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank component for 4-bit diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-Diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Zhiteng Li, Hanxuan Li, Junyi Wu, Kai Liu, Haotong Qin, Linghe Kong, Guihai Chen, Yulun Zhang, and Xiaokang Yang. DVD-Quant: Data-free video diffusion transformers quantization. In *International Conference on Learning Representations (ICLR)*, 2026.

Chengxi Liao and Zeyi Wen. Channel-aware mixed-precision quantization for efficient long-context inference. In *International Conference on Learning Representations (ICLR)*, 2026.

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*, pp. 32332–32344. PMLR, 2024.

Jiayi Luo, Jiayu Chen, Jiankun Wang, Cong Wang, Hanxin Zhu, Qingyun Sun, Chen Gao, Zhibo Chen, and Jianxin Li. Attention sparsity is input-stable: Training-free sparse attention for video generation via offline sparsity profiling and online QK co-clustering. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Xiaofeng Mao, Shaoheng Lin, Zhen Li, Chuanhao Li, Wenshuo Peng, Tong He, Jiangmiao Pang, Mingmin Chi, Yu Qiao, and Kaipeng Zhang. Yume: An interactive world generation model, 2025.

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, and Tong Zhang. Longcat-video technical report, 2025.

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

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, 2024.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

J. Shin, Z. Li, R. Zhang, J.-Y. Zhu, J. Park, E. Shechtman, and X. Huang. Motionstream: Real-time video generation with interactive motion controls. In *International Conference on Learning Representations (ICLR)*, 2026.

Z. Su, H. Wei, Z. Chen, W. Shen, L. Li, H. Yu, and K. Yuan. RotateKV: Accurate and robust 2-bit KV cache quantization for LLMs via outlier-aware adaptive rotations. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 6200–6208, 2025.

W. Sun, H. Zhang, H. Wang, J. Wu, Z. Wang, Z. Wang, Y. Wang, J. Zhang, T. Wang, and C. Guo. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Siao Tang, Xinyin Ma, Gongfan Fang, Xingyi Yang, and Xinchao Wang. Q-ARVD: Quantizing autoregressive video diffusion models, 2026.

Qian Tao, Wenyuan Yu, and Jingren Zhou. AsymKV: Enabling 1-bit quantization of KV cache with layer-wise asymmetric quantization configurations, 2024. URL https://arxiv.org/abs/2410.13212.

Team Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, et al. Wan: Open and advanced large-scale video generative models, 2025.

Tuna Tuncer, Felix Becker, and Thomas Pfeil. Quantized keys steal attention: Bias correction for KV-cache compression in video diffusion, 2026. URL https://arxiv.org/abs/2605.26266.

Haocheng Xi et al. Quant videogen: Auto-regressive long video generation via 2-bit KV-cache quantization. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable KV cache compression via importance-aware mixed precision quantization, 2024. URL https://arxiv.org/abs/2402.18096.

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

T. Yin, Q. Zhang, R. Zhang, W. T. Freeman, F. Durand, E. Shechtman, and X. Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22963–22974, 2025.

Tao Zhang, Ziqian Zeng, Hao Peng, Huiping Zhuang, and Cen Chen. MixKVQ: Query-aware mixed-precision KV cache quantization for long-context reasoning. *arXiv preprint arXiv:2512.19206*, 2025.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Rui Wan, Widyadewi Soedarmadji, Enshu Liu, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. ViDiT-Q: Efficient and accurate quantization of diffusion transformers for image and video generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal Forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *arXiv preprint arXiv:2602.02214*, 2026.

A SPATIOTEMPORAL HADAMARD SMOOTHING

This section expands Section 3.1 using the same token-block notation. For $K_{b,i}, V_{b,i} \in \mathbb{R}^{h\times C}$ and $h = 128$, the normalized Sylvester Hadamard matrix satisfies

$$H_h^\top H_h = hI, \qquad W_h = \frac{H_h}{\sqrt{h}}, \qquad W_h^\top W_h = I, \qquad W_h^\top = W_h. \tag{6}$$

Left multiplication acts along tokens independently for each channel. The smoothed blocks therefore preserve Frobenius energy:

$$\|(K_s)_{b,i}\|_F^2 = \|K_{b,i}\|_F^2, \qquad \|(V_s)_{b,i}\|_F^2 = \|V_{b,i}\|_F^2. \tag{7}$$

Define the quantization errors in the smoothed domain as

$$\begin{aligned} E_{K,b,i} &= Q^{-1}(Q((K_s)_{b,i})) - (K_s)_{b,i}, \\ E_{V,b,i} &= Q^{-1}(Q((V_s)_{b,i})) - (V_s)_{b,i}. \end{aligned} \tag{8}$$

Applying the inverse transform yields

$$\begin{aligned} \widehat{K}_{b,i} - K_{b,i} &= W_h^\top E_{K,b,i}, \quad \|\widehat{K}_{b,i} - K_{b,i}\|_F^2 = \|E_{K,b,i}\|_F^2, \\ \widehat{V}_{b,i} - V_{b,i} &= W_h^\top E_{V,b,i}, \quad \|\widehat{V}_{b,i} - V_{b,i}\|_F^2 = \|E_{V,b,i}\|_F^2. \end{aligned} \tag{9}$$

Thus, the inverse transform does not amplify the quantization error in Frobenius norm. Smoothing changes the distribution presented to the quantizer rather than reducing the tensor's energy. Orthogonality alone does not guarantee a smaller quantization error for every input. These identities hold in exact arithmetic, with finite-precision computation introducing additional rounding.

Incomplete token blocks. For a compression span with S tokens, let $r = S \bmod 128$. If $r > 0$, the final Key and Value blocks contain r valid tokens. We append $128 - r$ zero rows and retain all 128 transformed rows through quantization, packing, and dequantization. The valid length is stored with the span. Cropping is performed only after the inverse Hadamard transform, so padded positions never become additional attention keys or values. Let $P_r \in \mathbb{R}^{128\times r}$ append zero rows, with $P_r^\top P_r = I_r$. The tail reconstruction is

$$\begin{aligned} \widehat{K}_{\text{tail}} &= P_r^\top W_h^\top Q^{-1}(Q(W_h P_r K_{\text{tail}}))\,, \\ \widehat{V}_{\text{tail}} &= P_r^\top W_h^\top Q^{-1}(Q(W_h P_r V_{\text{tail}}))\,. \end{aligned} \tag{10}$$

Without quantization, $P_r^\top W_h^\top W_h P_r = I_r$, so the valid tokens are recovered exactly in exact arithmetic. If E_K and E_V denote quantization errors in the padded transform domain, then $\|\widehat{K}_{\text{tail}} - K_{\text{tail}}\|_F \le \|E_K\|_F$ and $\|\widehat{V}_{\text{tail}} - V_{\text{tail}}\|_F \le \|E_V\|_F$. No padding is needed when $r = 0$.

B OFFLINE LAYER-WISE PRECISION BUDGETS

B.1 LAYER GROUPING

Each layer has an offline sparsity score $S_l = 1 - D_l$, where density uses the 80% attention-mass threshold in Section 3.2. The three-group K-means partition is

$$\min_{\{G_j,\mu_j\}_{j=1}^3} \sum_{j=1}^{3} \sum_{l\in G_j} (S_l - \mu_j)^2, \qquad \mu_j = \frac{1}{|G_j|} \sum_{l\in G_j} S_l. \tag{11}$$

Ordering the cluster means as $\mu_1 \le \mu_2 \le \mu_3$ gives $L_R = G_1$, $L_N = G_2$, and $L_S = G_3$. These assignments fix channel counts at each precision, while the identities of the channels are selected at runtime.

B.2 Normal-Layer Budget

In HiMix-low, the high-, medium-, and low-precision groups use 4, 2, and 1 bits. The normal-layer counts satisfy

$$\begin{aligned} C_H^N + C_M^N + C_L^N &= C_{\text{total}}^N, \\ 4C_H^N + 2C_M^N + C_L^N &= 2C_{\text{total}}^N. \end{aligned} \tag{12}$$

Subtracting twice the first equality from the second gives $2C_H^N = C_L^N$. Substituting $C_L^N = \alpha C_{\text{total}}^N$ yields

$$(C_H^N, C_M^N, C_L^N) = C_{\text{total}}^N \left(\frac{\alpha}{2}, 1 - \frac{3\alpha}{2}, \alpha \right), \tag{13}$$

recovering Eq. (4). Nonnegative counts require $0 \le \alpha \le 2/3$. At $\alpha = 0.6$, the fractions are 0.3, 0.1, and 0.6, respectively. The ablation selects α, whereas the payload constraint determines the factor $1/2$ in the high-precision fraction.

B.3 Sensitive and Robust Layers

For layers with the same total channel count, sensitive layers halve the normal low-precision count and retain the high-precision count:

$$C_L^S = \frac{C_L^N}{2}, \qquad (C_H^S, C_M^S, C_L^S) = C_{\text{total}}^S \left(\frac{\alpha}{2}, 1 - \alpha, \frac{\alpha}{2} \right). \tag{14}$$

Robust layers halve the normal high-precision count and retain the low-precision count:

$$C_H^R = \frac{C_H^N}{2}, \qquad (C_H^R, C_M^R, C_L^R) = C_{\text{total}}^R \left(\frac{\alpha}{4}, 1 - \frac{5\alpha}{4}, \alpha \right). \tag{15}$$

In both cases the remaining channels use medium precision. For $g \in \{N, S, R\}$, define $\bar{b}_K^g = (4C_H^g + 2C_M^g + C_L^g)/C_{\text{total}}^g$. Then

$$\bar{b}_K^N = 2, \qquad \bar{b}_K^S = 2 + \frac{\alpha}{2}, \qquad \bar{b}_K^R = 2 - \frac{\alpha}{2}. \tag{16}$$

At $\alpha = 0.6$, these averages are 2, 2.3, and 1.7 bits. For L equally sized layers with $k = |L_S|$ and $r = |L_R|$,

$$\bar{b}_K = 2 + \frac{\alpha(k-r)}{2L}, \qquad \bar{b}_{KV} = \frac{\bar{b}_K + 2}{2} = 2 + \frac{\alpha(k-r)}{4L}. \tag{17}$$

The second expression assumes equally sized Key and Value caches and uniform 2-bit Value quantization. HiMix-high preserves the same channel fractions and doubles the precision of both caches. These are ideal quantized payload averages before integer rounding. They exclude scales, zero-points, channel permutations, packing overhead, and the recent BF16 cache, so they do not directly determine the measured memory compression ratio. The model-wide budget is approximately 2 bits in HiMix-low rather than an enforced exact equality.

C Runtime Channel Saliency

C.1 Query-Dependent Logit Perturbation

For one attention head, let $z_{t,j} = q_t^\top k_j / \sqrt{d}$. Holding Query fixed and writing $\widehat{k}_j = k_j + \delta k_j$ gives

$$\delta z_{t,j} = \frac{1}{\sqrt{d}} \sum_{c=1}^{d} q_{t,c} \delta k_{j,c}, \qquad |\delta z_{t,j}| \le \frac{1}{\sqrt{d}} \sum_{c=1}^{d} |q_{t,c}| \, |\delta k_{j,c}|. \tag{18}$$

A channel's contribution therefore depends on both its Key reconstruction error and its Query magnitude, not on the Key range alone.

For intuition, consider direct uniform scalar quantization over a channel range ΔK_c with b bits, nearest rounding, and no clipping. The step size is $\Delta K_c/(2^b - 1)$, giving the channel-wise bound

$$|\delta z_{t,j}^{(c)}| \le \frac{|q_{t,c}| \Delta K_c}{2\sqrt{d}(2^b - 1)}. \tag{19}$$

Averaging the numerator over S query tokens gives

$$s_c = \frac{1}{S}\sum_{t=1}^{S} |Q_{t,c}|\Delta K_c, \tag{20}$$

which recovers Eq. (5). At a common reference bit-width, the omitted factors do not change the channel ranking.

Equation (19) motivates the score under direct scalar quantization. It is not an error bound for the full HiMix pipeline, which scores raw Query and Key before RoPE and quantizes Hadamard-smoothed tensors. Softmax and Value activations also affect the attention-output perturbation. The score is an importance proxy, not an exact output-error prediction.

C.2 Assignments Within Fixed Budgets

Let π_l order the C^l_{total} channels of layer l by decreasing s_c. The three channel sets are

$$\begin{aligned} \mathcal{C}^l_H &= \{\pi_l(1), \ldots, \pi_l(C^l_H)\}, \\ \mathcal{C}^l_M &= \{\pi_l(C^l_H + 1), \ldots, \pi_l(C^l_H + C^l_M)\}, \\ \mathcal{C}^l_L &= \{\pi_l(C^l_H + C^l_M + 1), \ldots, \pi_l(C^l_{\text{total}})\}. \end{aligned} \tag{21}$$

The counts remain fixed, while the permutation adapts to the most recently captured Query and Key. Ranking spans the flattened head channels. Each Key group uses per-channel quantization, while Value uses per-token quantization at 2 bits in HiMix-low and 4 bits in HiMix-high.

D Online KV Cache Processing

Algorithm 1 uses fixed budgets $B_l = (C^l_H, C^l_M, C^l_L)$ and mixed BF16/packed caches $\mathcal{C}_l$. J is the number of chunks and F the compression interval. Chunk indices start at zero. $\mathcal{T}_j$ includes denoising evaluations followed by the context-cache refresh. Captured raw tensors $(Q^\star_l, K^\star_l)$ supply saliency at the next trigger. Only the new span $I = [j - F, j)$ is compressed. $\mathrm{Assign}(\pi, B_l, \mathbf{b})$ assigns bit-widths in descending saliency order. $\mathcal{H}$ is the normalized token-axis Hadamard transform with block size 128. $\mathcal{P}^{\mathrm{ch/tok}}_b$ denotes asymmetric per-channel/per-token quantization and uint8 packing, including scales and zero-points. $\boldsymbol{\beta}$ follows the reordered channel layout.

Read reconstructs each packed span as

$$\widehat{K} = \mathcal{H}^{-1}([\mathcal{D}(P_K)]_{:,\pi^{-1}}), \qquad \widehat{V} = \mathcal{H}^{-1}(\mathcal{D}(P_V)), \tag{22}$$

where $\mathcal{D}$ unpacks and dequantizes. It assembles these spans with BF16 entries in token order. For a padded span, the stored valid length is used to crop after inverse smoothing as in Eq. (10), before concatenation. Older packed spans retain their assignments. $X^l_{j,\tau}$ is the layer-l hidden state. Step_l denotes the remaining operations after attention. RoPE uses the original query/key positions.

Algorithm 1 Online KV cache processing in HiMix

Require: $\{B_l\}_{l=1}^{L}$, F, J, $m \in \{1, 2\}$ (low/high mode)
$\mathcal{C}_l \leftarrow \varnothing$ $(l = 1{:}L)$, $\mathbf{b} \leftarrow m(4, 2, 1)$, $b_V \leftarrow 2m$
for $j = 0, \ldots, J-1$ **do**
  if $j \geq F \ \wedge \ j \bmod F = 0$ **then**
    $I \leftarrow [j - F, j)$
    for $l = 1, \ldots, L$ **do**
      $(K, V) \leftarrow \mathcal{C}_l[I]$
      $\mathbf{s} \leftarrow \mathrm{mean}_t\, |Q_l^\star| \odot (\max_t K_l^\star - \min_t K_l^\star)$
      $\pi \leftarrow \mathrm{argsort}_\downarrow(\mathbf{s})$, $\boldsymbol{\beta} \leftarrow \mathrm{Assign}(\pi, B_l, \mathbf{b})$
      $P_K \leftarrow \mathcal{P}^{\mathrm{ch}}_{\boldsymbol{\beta}}([\mathcal{H}(K)]_{:,\pi})$, $P_V \leftarrow \mathcal{P}^{\mathrm{tok}}_{b_V}(\mathcal{H}(V))$
      $\mathcal{C}_l[I] \leftarrow (P_K, P_V, \pi, I)$
    end for
  end if
  for $\tau \in \mathcal{T}_j$ **do**
    for $l = 1, \ldots, L$ **do**
      $(Q, K, V) \leftarrow \mathrm{QKV}_l(X_{j,\tau}^{l-1})$, $(Q_l^\star, K_l^\star) \leftarrow (Q, K)$
      $\mathcal{C}_l[j] \leftarrow (K, V)_{\mathrm{BF16}}$, $(\widehat{K}, \widehat{V}) \leftarrow \mathrm{Read}(\mathcal{C}_l, 0{:}j)$
      $(Q_R, K_R) \leftarrow (\mathrm{RoPE}(Q), \mathrm{RoPE}(\widehat{K}))$
      $O \leftarrow \mathrm{softmax}(Q_R K_R^\top / \sqrt{d})\widehat{V}$
      $X_{j,\tau}^{l} \leftarrow \mathrm{Step}_l(X_{j,\tau}^{l-1}, O)$
    end for
  end for
end for

E Supplementary Measurements and Ablations

E.1 Layer-Sensitivity Measurements

Controlled attention-output comparison. We collect Query, Key, and Value along an unmodified BF16 inference trajectory. Each measurement corresponds to one head at a particular layer, video chunk, and denoising timestep. Query rows are sampled deterministically, and all visible keys are retained. Key-only and Value-only diagnostics quantize a separate copy of the historical cache using per-channel Key and per-token Value quantization, respectively. Recent entries remain in their original precision. The live cache and subsequent inference states are unchanged, so these diagnostic perturbations do not propagate.

For one measurement, let Q and K denote the tensors used in the attention dot product, including positional encoding where applicable. With $\widehat{K}$ and $\widehat{V}$ denoting the reconstructed cache,

$$\begin{aligned} &A = \mathrm{softmax}(QK^\top/\sqrt{d}), \qquad \widehat{A} = \mathrm{softmax}(Q\widehat{K}^\top/\sqrt{d}), \\ &O = AV, \qquad\qquad\qquad \widehat{O}_K = \widehat{A}V, \qquad \widehat{O}_V = A\widehat{V}. \end{aligned} \tag{23}$$

These comparisons isolate local output perturbations. They do not measure accumulated final-video degradation or evaluate the full mixed-precision HiMix pipeline.

Layer-wise error aggregation. Let $\mathcal{M}_l$ collect measurements for layer l and $E_m = \|O_m\|_F^2$ denote baseline output energy. For $u \in \{K, V\}$, the pooled error is

$$\varepsilon_l^{(u)} = \left(\frac{\sum_{m \in \mathcal{M}_l} \|\widehat{O}_m^{(u)} - O_m\|_F^2}{\sum_{m \in \mathcal{M}_l} E_m} \right)^{1/2}. \tag{24}$$

This is Eq. (3) applied to concatenated outputs, not the arithmetic mean of individual relative errors. Per-prompt estimates include only that prompt. Pooled estimates accumulate both the numerator and denominator across prompts, heads, chunks, and denoising timesteps.

Attention density and sparsity. For a sampled query t in measurement m, sort the normalized attention weights as $a_{m,t,(1)} \geq \cdots \geq a_{m,t,(n_m)}$, where n_m is the number of visible keys. With

$\tau = 0.8$, define

$$k_{m,t} = \min\left\{ k : \sum_{j=1}^{k} a_{m,t,(j)} \geq \tau \right\}, \qquad D_m = \frac{1}{|\mathcal{Q}_m|} \sum_{t \in \mathcal{Q}_m} \frac{k_{m,t}}{n_m}. \tag{25}$$

Here $\mathcal{Q}_m$ contains the sampled query positions. The plotted layer score uses the same baseline energy as the error aggregation:

$$D_l = \frac{\sum_{m \in \mathcal{M}_l} E_m D_m}{\sum_{m \in \mathcal{M}_l} E_m}, \qquad S_l = 1 - D_l. \tag{26}$$

Higher sparsity indicates more concentrated attention. The 80% threshold and $S = 1 - D$ convention match Section 3.2.

Ranking comparisons. Writing $R(x)$ for the rank vector, with average ranks for ties, Spearman correlation is

$$\rho_s(x, y) = \mathrm{Corr}(R(x), R(y)). \tag{27}$$

Figure 3(a) compares layer rankings between every pair of ten prompts, giving $\binom{10}{2} = 45$ correlations for each metric. These pairs share prompts and are not 45 independent prompt samples. Figure 3(b) compares pooled layer-wise sparsity and Key-only error under 2-bit quantization.

Figure 3(c) displays ten selected illustrative prompts from the bit-width experiments, rather than the same prompt set as Figure 3(a). For displayed prompt p and bit-width b, define

$$\rho_p^{(b)} = \rho_s\Big((S_{l,p})_{l=1}^{L}, (\varepsilon_{l,p}^{(K,b)})_{l=1}^{L} \Big), \qquad \Delta_\rho = \frac{1}{P} \sum_{p=1}^{P} |\rho_p^{(2)} - \rho_p^{(4)}|, \quad P = 10. \tag{28}$$

The mean absolute change summarizes the selected examples, not all tested prompts. It assesses stability of the sparsity–error association, not equality of error magnitudes or identical error rankings across precisions. These empirical comparisons do not establish a quantization-independent sensitivity bound.

E.2 Channel-Saliency Measurements

Top-10% channel sets. For one head with dimension d, let $k_{10} = \lceil 0.1d \rceil$. For a query token t, the diagnostic sets and miss rate are

$$\begin{aligned} \mathcal{T}_K &= \mathrm{Top}_{k_{10}}\{\Delta K_c\}_{c=1}^{d}, \\ \mathcal{T}_{QK}(t) &= \mathrm{Top}_{k_{10}}\{|Q_{t,c}|\Delta K_c\}_{c=1}^{d}, \\ M(t) &= 1 - \frac{|\mathcal{T}_K \cap \mathcal{T}_{QK}(t)|}{k_{10}}. \end{aligned} \tag{29}$$

For each saved layer and token block, we compute the max–min key range over all tokens separately for each head and channel. The K-only ranking uses this range, whereas the joint ranking multiplies it by the absolute query activation at the sampled token. Both rankings select the same number of channels: the ceiling of 10% of the head dimension. With 128 channels per head, each set contains 13 channels. The definition of $\mathrm{MissRate}_{10\%}$ in the main text therefore counts the joint top-13 channels absent from the K-only top-13 set and divides by 13. A value of zero means identical selections. A value of one means no overlap. The denominator is the joint top-channel set, not all 128 channels.

Sampling and aggregation. The saved tensors cover 30 layers, 12 heads, and 13 token blocks from one inference trajectory. Each block contains 4,680 tokens. We select 16 evenly spaced query positions, including the endpoints, with positions rounded to integer indices. This gives 74,880 token–head comparisons. Each heatmap cell in Figure 4(b) is the arithmetic mean of the miss rates over the 208 sampled token positions across blocks for one layer–head pair. Writing $\mathcal{Q}_v$ for the 16 sampled positions in block v, the value for layer l and head a is

$$\overline{M}_{l,a} = \frac{1}{13 \cdot 16} \sum_{v=1}^{13} \sum_{t \in \mathcal{Q}_v} M_{l,a,v}(t). \tag{30}$$

The overall mean is 66.0%. These measurements characterize variation across layers, heads, and token positions within the captured trajectory. They do not establish cross-prompt robustness.

Relation to runtime saliency. The diagnostic uses an individual query token to reveal query-dependent ranking changes. Runtime allocation instead averages absolute query activations over the token block, as in Eq. (5), and ranks channels across the flattened head dimensions to assign a shared precision layout. The representative example in Figure 4(a) uses layer 1, head 1, and token 100 of the first block. Its orange bars highlight the three highest-scoring missed channels under the joint criterion, rather than all missed channels.

Matching the runtime ranking scope. To test whether the per-head diagnostic transfers to layer-wide allocation, we repeat the comparison on all 390 saved layer–block pairs from the same trajectory. A matched 2×2 comparison varies the ranking domain (128 channels within each head or 1,536 flattened channels within each layer) and the Query statistic (an individual token or the mean absolute activation over all 4,680 tokens in the block). The individual-token setting retains the same 16 sampled positions as Figure 4. Each comparison selects $\lceil 0.1C \rceil$ channels, giving 13 per head or 154 per layer. For the layer-wide, block-mean setting,

$$\begin{aligned} \mathcal{T}_K^{\text{layer}} &= \text{Top}_{154}\{\Delta K_c\}_{c=1}^{1536}, \\ \mathcal{T}_{QK}^{\text{layer}} &= \text{Top}_{154}\left\{\left(\frac{1}{4680}\sum_{t=1}^{4680}|Q_{t,c}|\right)\Delta K_c\right\}_{c=1}^{1536}, \\ M^{\text{layer}} &= 1 - \frac{|\mathcal{T}_K^{\text{layer}} \cap \mathcal{T}_{QK}^{\text{layer}}|}{154}. \end{aligned} \tag{31}$$

This uses the same scoring rule and ranking domain as runtime allocation. The 10% selection is retained for diagnostic comparison, rather than substituted for the offline high-, medium-, and low-precision budgets.

Table 7: Top-10% miss rate under matched ranking scopes and Query statistics. All settings use the same 30 layers and 13 blocks from one trajectory. The range is the minimum and maximum of the 30 layer means.

Ranking scope	Query statistic	Mean (%)	Layer range (%)
Per head	Individual token	66.04	48.64–74.15
Whole layer	Individual token	61.90	47.86–70.06
Per head	Block mean	57.95	49.36–68.64
Whole layer	Block mean	50.60	30.52–63.99

We first average within each layer over blocks and, where applicable, sampled tokens and heads, then average equally over all 30 layers. Table 7 reproduces the original per-head result. Layer-wide ranking and Query averaging reduce the miss rate, but the runtime-matched setting still omits 50.60% of the jointly selected channels on average. Every layer has a nonzero mean miss rate, and 18 of 30 layers exceed 50%. Thus, the discrepancy between Key-only and joint ranking persists when channels compete across heads using block-averaged Query activations, rather than being confined to the per-head diagnostic.

E.3 Low-Precision Fraction and Channel Statistics

We report low-magnitude Key channel statistics as supplementary descriptive evidence, separate from the layer-wise sparsity clustering in Appendix B. Across 30 layers and five calibration prompts (150 layer–prompt pairs), the mean number of channels in this cluster, K_{low}, is 998.3 out of 1,536, or 65.0%. The five prompt-level means range from 985.1 to 1,012.0 channels (64.1%–65.9%), with a coefficient of variation of 0.94%.

The layer-wise curves retain a similar overall shape despite substantial cross-layer heterogeneity (Figure 6). The per-layer mean ranges from 426.4 channels in layer 18 (27.8%) to 1,530.8 in layer 29 (99.7%). The cross-prompt coefficient of variation is below 5% for 26 of the 30 layers and below

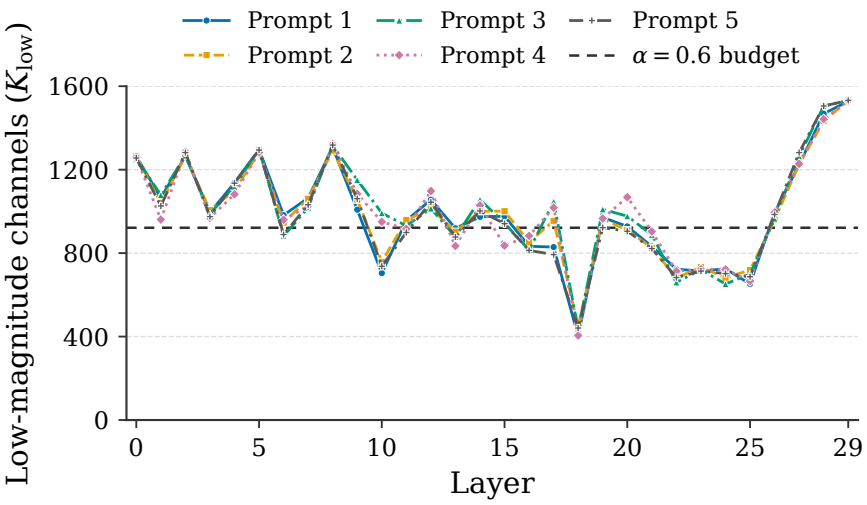


Figure 6: Number of K-cache channels assigned to the low-magnitude cluster (K_{low}) in each transformer layer for five calibration prompts. Each curve corresponds to one prompt. Every layer contains 1,536 channels. These channel statistics are distinct from the layer-wise sparsity clustering.

10% for 28 layers. Layers 10 and 17 show the largest prompt-dependent variation, with little effect on the prompt-level mean. These observations motivate a shared reference budget with layer-specific adjustments.

The observed low-magnitude fraction of approximately 65% motivates exploring reference budgets with a substantial low-precision group. We select $\alpha = 0.6$ using the Image Quality ablation in Table 5, where it performs best among $\{0.2, 0.4, 0.6, 0.66\}$. This assigns 60% of channels to 1-bit precision in normal layers and leaves 40% at medium or high precision. The cluster statistics provide



Figure 7: Additional qualitative comparison in a forest scene. Rows show BF16, KIVI with 2-bit KV cache quantization, QVG with 2-bit KV cache quantization, and HiMix-low, from top to bottom. Columns show successive frames from each generated sequence.

motivation rather than determine an optimal value. They do not imply that every low-magnitude channel tolerates 1-bit quantization.

The deployed channel identities are determined by joint Query–Key saliency under the offline layer budgets, not by this magnitude clustering. The choice of α is empirical rather than a theoretical optimum.

E.4 Long-Video Benchmark Protocol

The long-video benchmark in Table 2 uses the first 120 prompts, in their original order, from the 1,003-prompt extended MovieGen evaluation set. All compared methods generate 30-second videos with Self-Forcing using this same prompt subset. Image Quality and Subject Consistency are evaluated with VBench.

E.5 Additional Qualitative Comparison

Figure 7 provides an additional visual comparison of generated frames, showing changes in subject appearance and scene detail over the displayed sequence.

F LLM Usage Statement

We acknowledge the use of large language models (LLMs), such as OpenAI's GPT-6 Astra, as writing assistants. Their role was strictly limited to proofreading and rephrasing sentences to improve language quality. They did not contribute to the research conception or experimental results.