FRAMEFOLD: RESPONSE-GUIDED KV CACHE CONSOLIDATION FOR AUTOREGRESSIVE LONG VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In autoregressive video generation, retaining all historical key–value (KV) states increases memory usage and per-step attention costs as videos grow longer. A fixed cache budget bounds these costs but raises the challenge of preserving historical content that supports consistent, high-quality generation. Existing methods control cache size through history selection or consolidation, yet deciding what to retain and how to limit the loss of useful content remains a shared challenge. Our analysis shows that spatial queries depend on different historical content and that higher current attention coverage does not necessarily yield better long-video quality. We introduce FrameFold, a training-free KV cache consolidation method that compares the content retrieved by the same recent historical queries before and after merging to guide both representative construction and merge selection. Exploiting temporal redundancy between adjacent historical entries, FrameFold merges their KV states into shared representatives while retaining complete spatial grids. It fits value-merging coefficients in closed form by matching conditional attention responses, then selects the merge with the lowest remaining response distortion. Under matched KV cache budgets, experiments on Self Forcing and Causal Forcing at 30 and 60 seconds and on LongLive at 120 seconds demonstrate improved consistency and visual quality. Continuous eight-minute Self Forcing rollouts further show less quality degradation than the baselines.

1 INTRODUCTION

Autoregressive video generation models now synthesize high-quality short clips (Huang et al., 2025; Yin et al., 2025; Zhu et al., 2026). Although causal generation supports continued prediction of subsequent frames or chunks (Chen et al., 2024; 2025; Sand.ai et al., 2025; Yang et al., 2026), extrapolation beyond the training horizon often leads to subject and background inconsistencies and a decline in visual quality (Li et al., 2026a;b; Liu et al., 2026; Yi et al., 2026). Historical keys and values (KV) provide context for consistent subsequent generation (Chen et al., 2026b; Hu et al., 2026; Zhu et al., 2025), but retaining the full history makes storage and per-step attention costs grow with video length (Lu et al., 2026). Under a fixed cache budget, the model must repeatedly discard or compress history, determining what information remains available to later generation (Kim et al., 2026; Lu et al., 2026; Yi et al., 2026; Zhao et al., 2026a). We study how to organize and compress useful history within this budget to sustain long-video consistency and visual quality.

Existing methods manage limited KV caches through token selection and memory consolidation. Deep Forcing retains historical tokens based on attention salience (Yi et al., 2026), potentially discarding cues needed for subsequent generation. FlowCache improves coverage through redundancy-aware scoring (Ma et al., 2026), while Future Forcing uses future-query proxies and merges values from some evicted tokens into retained tokens to reduce information loss (Luo et al., 2026). For memory consolidation, MemRoPE updates memories through exponential moving averages (Kim et al., 2026), while FadeMem merges adjacent historical entries according to temporal distance (Lu et al., 2026), extending temporal coverage within a fixed budget. However, both methods use preset update rates or temporal structure to determine update weights or merge choices, without directly evaluating the resulting content loss. Token selection and memory consolidation thus share a chal-

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.

lenge: assessing the value of history for subsequent generation to decide **what to retain and how to retain it.**

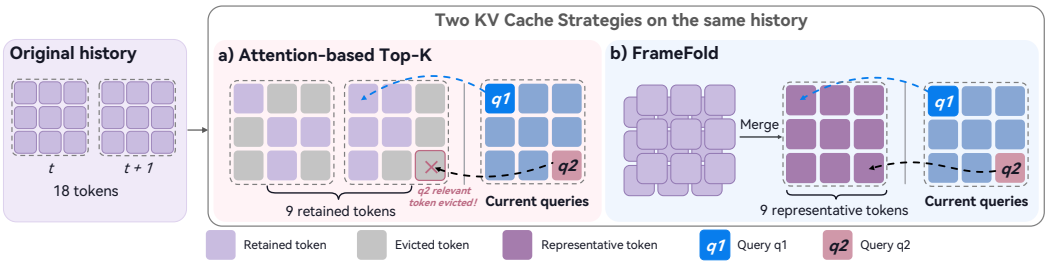

Figure 1: **Token selection and shared representation.** (a) Attention-based Top-K selection retains subsets of historical tokens; (b) FrameFold consolidates adjacent history into a representative grid. Actual Deep Forcing selection maps are shown in Appendix A.1.

Our analysis shows that spatial queries depend on different historical content, and a shared Top-K ranking based on aggregated attention can omit information important to some queries (Figure 2(a)). At a matched candidate retention ratio, replacing Deep Forcing's Top-K rule with Bottom-K reduces current attention coverage yet improves the evaluated consistency and visual-quality metrics on two generators, with lower dynamic degree (Figure 2(b)). This reversal shows that current attention coverage alone is insufficient to assess history retention. To meet different queries' historical needs within a fixed budget, we exploit temporal redundancy in neighboring frames and their features (Ji et al., 2026; Lu et al., 2026; Samuel et al., 2026; Xi et al., 2026) to merge adjacent historical KV entries at corresponding spatial locations. Each representative retains a complete spatial grid while covering a longer historical interval. The key is to limit content loss so that shared representatives preserve as closely as possible what the original history supplies to different queries.

We introduce FrameFold, a training-free KV cache consolidation method that uses conditional-response preservation to guide representative construction and merge selection (Figure 1). Recent historical queries serve as probes to compare the content retrieved from the original history and its shared representative. Normalizing attention within each block separates this comparison from the history's current share of global attention. For each candidate pair, FrameFold merges keys by temporal span and fits the value-merging coefficient in closed form to match the original conditional response. It then executes the merge with the lowest remaining response distortion.

Under matched KV cache budgets, FrameFold achieves the highest consistency, motion smoothness, and visual-quality scores among the evaluated methods on Self Forcing and Causal Forcing at both 30 and 60 seconds. Experiments on LongLive at 120 seconds further demonstrate gains in consistency and visual quality. Continuous eight-minute Self Forcing rollouts show higher consistency and visual quality with smaller declines than those of FadeMem and Deep Forcing. Ablation studies support conditional-response matching as a consolidation objective and the contributions of representative construction and merge selection.

2 Related Work

Autoregressive Video Generation. Autoregressive video generation predicts frames or chunks sequentially, providing a basis for streaming synthesis (Chen et al., 2024; 2025; Sand.ai et al., 2025). Recent work improves generation quality through advances in training and distillation (Guo et al., 2026b; Huang et al., 2025; Yin et al., 2025; Zhu et al., 2026) and enhances long-horizon stability through training and rollout strategies (Cui et al., 2026; Li et al., 2026b; Liu et al., 2026; Yang et al., 2026). Autoregressive generation also supports interactive world modeling under action, camera, or text control (Gao et al., 2026; Huang et al., 2026a; Mao et al., 2025). To support consistent scene revisits, WorldPlay (Sun et al., 2026) retrieves and reorganizes historical context, while Matrix-Game 3.5 (Qian et al., 2026) uses geometry-aware patch memory. As generated sequences grow longer, retaining and organizing useful history within a limited KV cache becomes important for maintaining long-term consistency.

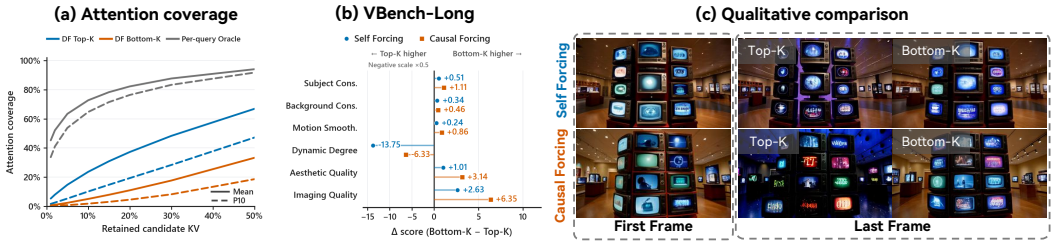

Figure 2: **Shared Top-K leaves query-level coverage gaps; higher coverage need not yield better generation quality.** **(a)** Candidate-region attention coverage for Top-K, Bottom-K, and a per-query oracle (solid: mean; dashed: P10). **(b)** VBench-Long score differences (Bottom-K minus Top-K). Despite lower coverage, Bottom-K improves five consistency and quality metrics on both models, with lower dynamic degree. **(c)** First-frame references and final frames for Self Forcing (top) and Causal Forcing (bottom). See Appendix A.2 for experimental details.

KV cache selection. KV cache selection determines which historical states to retain and how to allocate limited capacity. In language models, retention policies exploit historical attention contributions and the persistence of importance (Liu et al., 2023; Zhang et al., 2023), or preserve attention sinks and recent context (Xiao et al., 2024). Further work adapts retention to head-specific attention patterns (Ge et al., 2024; Li et al., 2024b) and allocates cache capacity across layers (Cai et al., 2025). In autoregressive video generation, selection criteria include attention-based importance (Yi et al., 2026), learned salience (Chen et al., 2026a), predicted future-query relevance (Luo et al., 2026), and temporal redundancy (Ma et al., 2026). Relax Forcing (Zhao et al., 2026b) balances global stability against redundancy relative to recent context, while TetherCache (Meng et al., 2026) combines attention relevance with temporal coverage of historical frames and aligns the statistics of recalled memory. Beyond selection criteria, retention and cache allocation adapt to the context needs of individual frames and attention heads (Cai et al., 2026; Guo et al., 2026a; Ji et al., 2026).

History Compression and Response Preservation. Complementary to selection, history compression encodes past information into compact representations. FramePack packs historical frames into compact context (Zhang et al., 2025), TinyHistory learns lightweight history embeddings (Zhang et al., 2026), and MAG (Zhu et al., 2025) and Echo-Infinity (Bian et al., 2026) aggregate history through learned memories. Representation merging offers another compression route: ToMe and VidToMe apply similarity-based token merging to vision transformers and video editing, respectively (Bolya et al., 2023; Li et al., 2024a). In language models, MiniCache exploits KV similarity between adjacent layers for compression across network depth (Liu et al., 2024), while KVMerger merges KV states according to key similarity (Wang et al., 2024). In autoregressive video generation, MemRoPE aggregates history through exponential moving averages (Kim et al., 2026), and FadeMem merges adjacent complete spatial grids (Lu et al., 2026). TempCache uses cross-frame correspondences for compression (Samuel et al., 2026), while Future Forcing guides merging with future-query projection profiles (Luo et al., 2026). Response preservation also guides representation construction: Attention Matching compacts language-model KV caches by matching local attention outputs and unnormalized attention mass (Zweiger et al., 2026). Like FadeMem, FrameFold merges adjacent complete spatial grids. It constructs shared representatives through conditional-response matching and selects merges by the remaining response distortion after fitting.

3 Motivation

3.1 Attention Coverage and Generation Quality

Queries at different spatial locations in a video often attend to different historical content. Even when an individual query concentrates its attention on a small subset of historical tokens, the subsets required by different queries need not coincide (Cai et al., 2026; Guo et al., 2026b). Under a fixed cache budget, Deep Forcing aggregates attention from multiple recent queries into an importance score for each historical token and retains the Top-K tokens according to this shared ranking (Yi et al., 2026) (Figure 1(a)). This policy serves different queries with the same retained subset: history

that is highly relevant to a few queries may rank below history that is broadly relevant to many queries and consequently be discarded.

We examine this issue through query-level attention coverage analysis (Figure 2(a)). At 50% candidate token retention, shared Top-K retains roughly two-thirds of the candidate-region attention on average, while its 10th-percentile coverage (P10) falls below one-half. By contrast, a per-query oracle that independently selects tokens for each query exceeds 90% in both mean coverage and P10 at the same retention ratio. The larger gap at P10 highlights the limited retention of history attended to by some queries. Visualizations of actual token selection also show uneven retention frequencies across spatial locations (Figure 7 in the appendix).

Higher attention coverage does not necessarily lead to better subsequent generation. We replace Top-K with Bottom-K while keeping the scoring procedure and candidate retention ratio unchanged. Despite retaining less current attention, Bottom-K achieves higher scores on all five consistency and quality metrics on both Self Forcing and Causal Forcing, with lower dynamic degree (Figure 2(b)). In the Causal Forcing example, Bottom-K also better preserves the initial scene's colors, whereas Top-K exhibits a pronounced blue cast (Figure 2(c)). These results show that lower-ranked history remains useful for generation. Compression decisions under a fixed budget should therefore consider **what historical content different queries can still retrieve after compression; current attention share alone is insufficient to assess the value of retaining that content.**

3.2 From Token Selection to Shared Representatives

To preserve history that different queries may need within a fixed budget, we consider allowing multiple historical entries to share cache slots. Temporal redundancy in video offers this opportunity: QVG (Xi et al., 2026) reports that latent tokens at corresponding spatial locations in adjacent frames often exhibit high similarity, suggesting that overlapping historical content can be stored through shared representatives. To accommodate the retrieval needs of queries at different spatial locations, we retain complete spatial grids and merge adjacent historical entries along the temporal dimension, combining KV features at corresponding spatial locations into a shared representative (Figure 1(b)). Each spatial location thus retains a historical representation, while a single grid represents a longer contiguous interval of history, reducing the number of stored entries. The remaining challenge is to control the content loss from merging so that the shared representative faithfully preserves what the original history supplies to queries.

4 Method

4.1 Evaluating Merges through Conditional Responses

History supplies content to queries through attention readouts: a query uses keys to form retrieval weights and aggregates values accordingly. We therefore assess content preservation by comparing the representations read by the same query before and after merging. In full-cache attention, a historical block's contribution to the output is also scaled by its share of global attention. To separate the block's readout from this share, we normalize attention within the block being evaluated and refer to the resulting weighted sum of values as its *conditional response*.

We measure the readout distortion introduced by merging as the difference between the conditional responses of the original history and its shared representative under the same query. Content preservation under this criterion leads to two coupled decisions: **which pair to merge** to minimize the resulting response distortion (Section 4.2); and **how to construct the shared representative** to match the original history's conditional responses as closely as possible (Section 4.3).

4.2 Distortion-Guided Merge Selection

When the KV cache exceeds its budget, we protect the initial sink entries and recent history and select adjacent entries in the intermediate candidate region for merging. Suppose this region currently contains n historical entries $\mathcal{F}_1, \ldots, \mathcal{F}_n$ in temporal order. There are $n-1$ adjacent candidate pairs, indexed by $\mathcal{E} = \{1, \ldots, n-1\}$, where candidate i corresponds to $(\mathcal{F}_i, \mathcal{F}_{i+1})$. Each iteration compares these candidates and executes one merge.

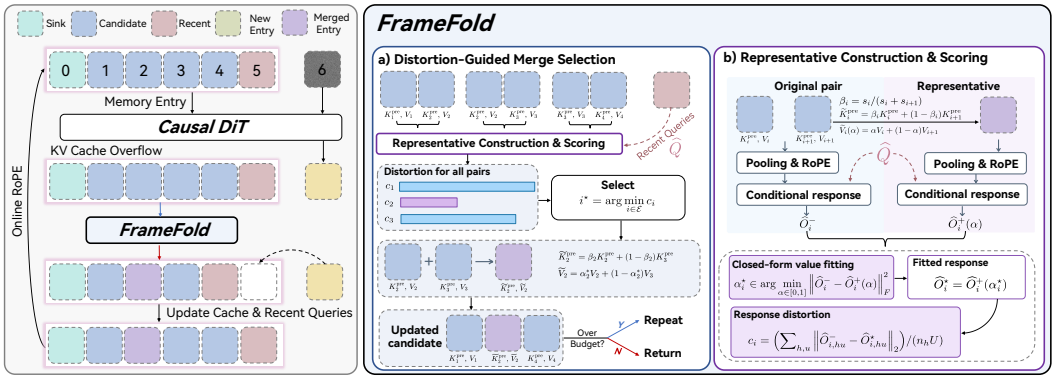

Figure 3: **Overview of FrameFold.** When the KV cache exceeds its budget, FrameFold consolidates adjacent entries in the candidate history while protecting sink and recent entries. Recent historical queries guide the evaluation of candidate merges, and the selected pair is replaced by a shared KV representative. Here, s_i denotes the number of original latent frames represented by entry i.

We use merge matrices to express the transformation from each candidate pair to its shared representative. Suppose each entry contains S spatial tokens. We denote the keys of entry $\mathcal{F}_i$ before rotary position encoding (RoPE) by K_i^{pre} and the corresponding values by V_i. Merging operates on the pre-RoPE keys and the corresponding values:

$$\widetilde{K}_i^{\mathrm{pre}} = M_i^K[K_i^{\mathrm{pre}}; K_{i+1}^{\mathrm{pre}}], \qquad \widetilde{V}_i = M_i^V[V_i; V_{i+1}], \tag{1}$$

where $M_i^K, M_i^V \in \mathbb{R}^{S\times 2S}$ map the two key and value grids, respectively, into one shared grid, and $[\,;\,]$ denotes concatenation along the token dimension. For each candidate, we first determine the merge matrices and construct a temporary representative using the rule in Section 4.3, and then evaluate its conditional-response distortion relative to the original pair.

For response evaluation, we denote queries after RoPE by Q. We obtain K_i by applying RoPE to K_i^{pre} at the entry's current cache positions, and $\widetilde{K}_i$ by applying RoPE to $\widetilde{K}_i^{\mathrm{pre}}$ at the representative's positions. Values do not receive RoPE. Hatted quantities denote evaluation representations derived from these QKV features; their construction is described in Section 4.3. All candidates share the same query probes $\widehat{Q}$. The original pair and its fitted representative have conditional responses

$$\begin{aligned}\widehat{O}_i^{-} &= \mathrm{softmax}\big(\widehat{Q}[\widehat{K}_i; \widehat{K}_{i+1}]^\top/\sqrt{d}\big)[\widehat{V}_i; \widehat{V}_{i+1}],\\ \widehat{O}_i^{\star} &= \mathrm{softmax}\big(\widehat{Q}\widehat{\widetilde{K}}_i^{\top}/\sqrt{d}\big)\widehat{\widetilde{V}}_i,\end{aligned} \tag{2}$$

where d is the attention head dimension. These responses describe what the same queries read from the original pair and the shared representative, with each softmax normalized within its respective historical block. We define the merge cost as their mean response difference:

$$c_i = \Big(\sum_{h,u}\Big\|\widehat{O}_{i,hu}^{-} - \widehat{O}_{i,hu}^{\star}\Big\|_2\Big)/(n_h U), \tag{3}$$

where the sum runs over n_h attention heads and U query probes, and each L_2 norm is taken over value channels. Every merge replaces two complete spatial grids with one, releasing the same cache capacity. We therefore select the candidate with the lowest cost:

$$i^\star = \arg\min_{i\in\mathcal{E}} c_i. \tag{4}$$

Each layer independently scores candidates and selects a pair shared by all its heads. We apply the selected merge matrices to the complete K^{pre} and V grids and replace the pair with a representative carrying their combined temporal span. We then reconstruct representatives and rescore all current candidates using the updated history, repeating until the cache meets its budget; query probes remain fixed throughout the update.

4.3 Response-Guided Representative Construction

We construct a shared representative for each candidate pair scored in Section 4.2. Given a fixed adjacent pair $(\mathcal{F}_i, \mathcal{F}_{i+1})$ and query probes, we determine its merge matrices so that the representative approximates the original conditional responses.

To preserve spatial organization, we form convex combinations of the two entries' features at corresponding locations. To keep fitting inexpensive, keys and values each use one coefficient shared across spatial locations and attention heads:

$$M_i^K = \big[\beta_i I_S,\ (1-\beta_i) I_S\big], \qquad M_i^V(\alpha) = \big[\alpha I_S,\ (1-\alpha) I_S\big], \tag{5}$$

where I_S is the identity matrix and $\beta_i, \alpha \in [0,1]$. We determine these two coefficients in turn.

Keys affect conditional responses nonlinearly through softmax. We first construct the shared keys $\widetilde{K}_i^{\mathrm{pre}}$ before RoPE (Su et al., 2024) using the span-weighted averaging rule of FadeMem (v1) (Lu et al., 2026). If entry $\mathcal{F}_i$ covers s_i original latent frames, we set $\beta_i = s_i/(s_i + s_{i+1})$, weighting the keys by the number of frames each entry represents.

With the representative's keys and positions fixed, its attention weights $A_i^+ = \mathrm{softmax}\big(\widehat{Q}\widehat{\widetilde{K}}_i^{\top}/\sqrt{d}\big)$ are independent of α. Its response is affine in α: $\widehat{O}_i^+(\alpha) = C_i + \alpha D_i$, where $C_i = A_i^+ \widehat{V}_{i+1}$ and $D_i = A_i^+(\widehat{V}_i - \widehat{V}_{i+1})$. We fit this candidate's coefficient by minimizing the squared response error against the original response $\widehat{O}_i^-$:

$$\alpha_i^\star \in \arg\min_{\alpha\in[0,1]} \left\| \widehat{O}_i^- - C_i - \alpha D_i \right\|_F^2 . \tag{6}$$

This objective is a scalar convex quadratic. For $\|D_i\|_F^2 > 0$, its interval-constrained optimum is

$$\alpha_i^\star = \mathrm{clip}_{[0,1]}\big(\langle D_i, \widehat{O}_i^- - C_i\rangle_F / \|D_i\|_F^2\big). \tag{7}$$

Under the same shared keys and query probes, the span-weighted coefficient $\alpha = \beta_i$ is feasible, so the fitted values incur no greater squared response error than span-weighted values. Substituting $\alpha_i^\star$ into M_i^V gives the candidate representative and its response $\widehat{O}_i^\star = \widehat{O}_i^+(\alpha_i^\star)$ for computing the merge cost in Section 4.2.

To reduce the cost of representative fitting and candidate scoring, we construct the hatted evaluation representations through pooling. We jointly pool saved queries from the latest R completed historical frames over time and space into U probes, and spatially pool each historical entry's keys and values to U tokens. Pooling operates on the position-encoded queries Q and keys K_i, and directly on values V_i. Fitting and scoring use these pooled representations, and the selected mixing coefficients are applied to the complete K^{pre} and V spatial grids. Additional position conventions and degenerate-case handling are given in Appendix B.1.

5 Experiments

5.1 Experimental setup

We evaluate FrameFold on pretrained Self Forcing and Causal Forcing models (Huang et al., 2025; Zhu et al., 2026) without additional training. The main comparison uses 128 MovieGenBench prompts (Polyak et al., 2024) refined with Qwen2.5-7B-Instruct, following the prompt setup of MemRoPE (Kim et al., 2026). We generate 30- and 60-second videos at 480×832 resolution and 16 fps. Baselines comprise the original backbones with FIFO cache eviction, Deep Forcing, and FadeMem (Lu et al., 2026; Yi et al., 2026). Within each generator–duration setting, methods share checkpoints, prompts, random seeds, and denoising schedules. All methods use a default budget of 12 frame-equivalent KV entries, including the three-frame current chunk, and protect three sink frames and the three most recent historical frames. Unless varied explicitly, FrameFold uses a spatial pooling factor of 4 and queries from the latest $R = 3$ completed historical frames for representative fitting and merge scoring.

Table 1: **Quantitative comparison of FrameFold with baselines on 30- and 60-second video generation.** We evaluate Self Forcing and Causal Forcing on 128 MovieGenBench prompts with a cache budget of 12 frame-equivalent KV entries. Higher is better; best scores within each backbone–duration group are in **bold**.

Config	Method	Subject Consist.↑	Background Consist.↑	Motion Smooth.↑	Dynamic Degree↑	Aesthetic Quality↑	Imaging Quality↑
30s	Self Forcing	96.38	95.89	97.77	59.11	57.40	66.80
	+ Deep Forcing	96.72	96.03	98.09	56.67	57.84	67.27
	+ FadeMem	96.94	96.15	98.08	56.35	58.01	68.29
	+ FrameFold	**97.16**	**96.29**	**98.25**	**60.42**	**58.27**	**68.52**
	Causal Forcing	95.65	95.52	96.95	75.73	55.80	66.31
	+ Deep Forcing	95.53	95.48	96.93	78.59	56.35	65.53
	+ FadeMem	95.73	95.44	97.08	**83.85**	57.08	68.08
	+ FrameFold	**95.82**	**95.53**	**97.17**	83.13	**57.55**	**68.91**
60s	Self Forcing	95.80	95.60	97.49	55.34	56.95	66.12
	+ Deep Forcing	96.39	95.89	98.02	57.59	57.19	67.14
	+ FadeMem	96.71	96.05	98.14	54.17	57.80	68.17
	+ FrameFold	**96.94**	**96.17**	**98.23**	**62.11**	**58.23**	**68.80**
	Causal Forcing	94.68	95.12	96.12	75.07	54.85	64.54
	+ Deep Forcing	95.45	95.16	96.85	73.67	55.64	64.87
	+ FadeMem	95.38	95.23	96.76	**85.31**	56.03	68.09
	+ FrameFold	**95.54**	**95.36**	**97.09**	83.36	**56.70**	**68.35**

We report the six VBench-Long dimensions (Huang et al., 2024; 2026b) supported for custom video evaluation: subject consistency and background consistency for temporal consistency; motion smoothness and dynamic degree for motion behavior; and aesthetic quality and imaging quality for visual quality. Scores are reported on a 0–100 scale.

5.2 Main results

FrameFold improves consistency and visual quality across both generators and rollout lengths under the same cache budget (Table 1). It achieves the best or tied-best subject consistency, background consistency, motion smoothness, aesthetic quality, and imaging quality in all four settings. At 60 seconds, aesthetic quality exceeds that of the strongest baseline by 0.43 points on Self Forcing and 0.67 points on Causal Forcing. Dynamic degree is highest on Self Forcing and lower than that of FadeMem on Causal Forcing, indicating that the quality gains accompany different motion levels across the two backbones. Appendix A.9 compares generation throughput at the shared 12-entry budget and additionally evaluates FIFO and Deep Forcing with 21 entries.

In the portrait example in Figure 4, FrameFold retains visible freckles and more stable skin coloration, while Deep Forcing loses fine facial texture and FIFO develops a stronger color cast. In the running example, Deep Forcing exhibits fragmented background textures in later frames, whereas FrameFold maintains cleaner scene structure and subject boundaries.

5.3 Long-horizon generation

We test sustained compression by generating continuous 480-second videos with Self Forcing on 24 prompts, using the same fixed cache budget. For each method, the evaluated 30-, 60-, 120-, 240-, and 480-second videos are nested prefixes of the same rollout.

Figure 5 shows higher subject and background consistency, aesthetic quality, and imaging quality for FrameFold than for FadeMem and Deep Forcing at all five durations. The decline from 30 to 480 seconds is also smaller in each of these four dimensions. At 480 seconds, FrameFold exceeds FadeMem by 1.40 points in aesthetic quality and 1.61 points in imaging quality. Complete six-dimensional results (Table 4) and temporal visualizations appear in Appendix A.4.

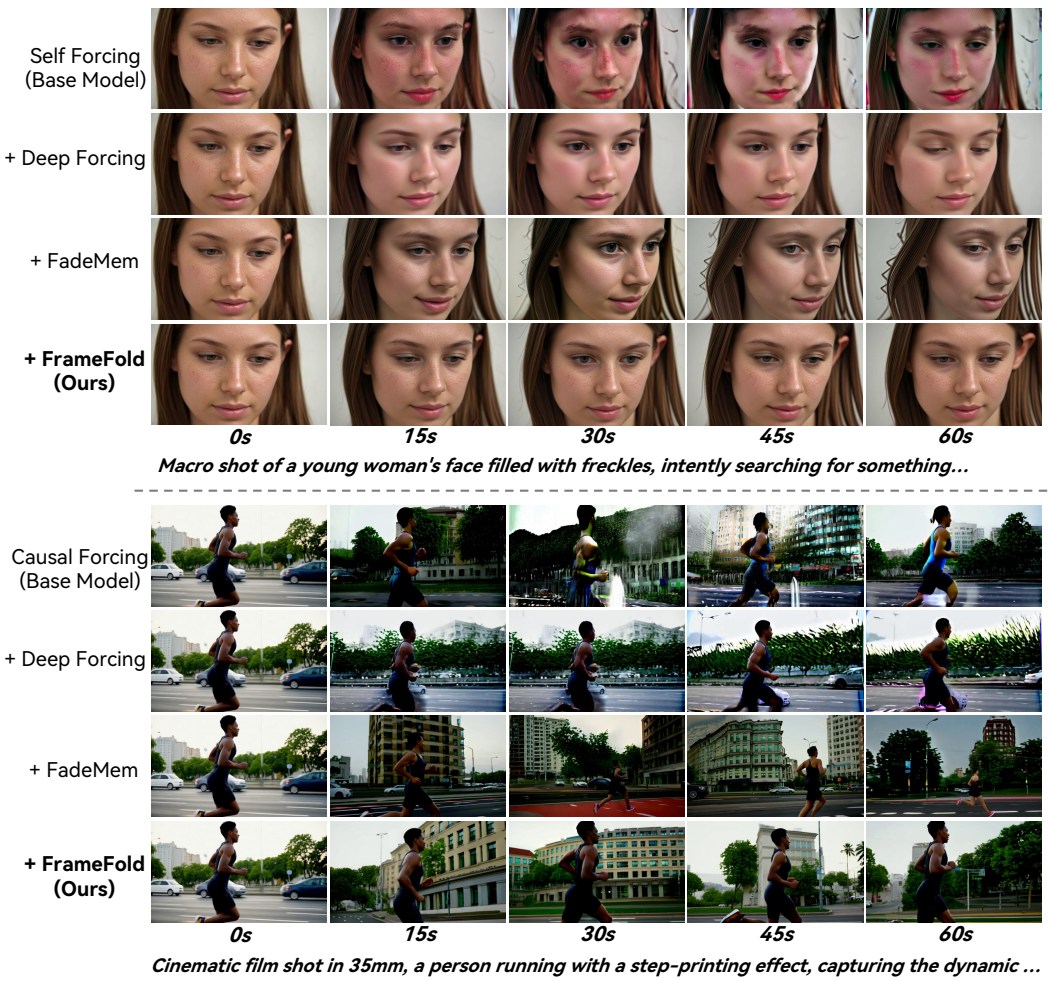

Figure 4: **Qualitative comparison of FrameFold with baselines on 60-second video generation.** Self Forcing and Causal Forcing examples are shown in the top and bottom groups, respectively. Methods within each example use the same prompt and seed, with a cache budget of 12 frame-equivalent KV entries.

5.4 Ablation Studies

Merge-selection and value-fitting ablations use 48 prompts, and the response-objective comparison uses 128 prompts. These studies use Self Forcing with 30-second videos and the default 12-entry cache budget.

Merge selection. With either Span or Optim, O diff achieves the highest subject and background consistency, motion smoothness, aesthetic quality, and imaging quality among the three selection criteria, with lower dynamic degree (Table 2).

Value fitting. Replacing Span with Optim improves subject and background consistency and motion smoothness under all three selection criteria. Imaging quality improves with K diff and O diff, with a 0.01-point decrease under V diff. With O diff, fitting increases subject consistency by 0.18 points and imaging quality by 0.13 points, with a 0.07-point decrease in aesthetic quality. Qualitative comparisons appear in Appendix A.5.

Response objective. Table 3 compares full-cache and conditional responses for both value fitting and merge selection. Conditional responses improve subject and background consistency, motion smoothness, aesthetic quality, and imaging quality, with a 0.72-point gain in aesthetic quality and a

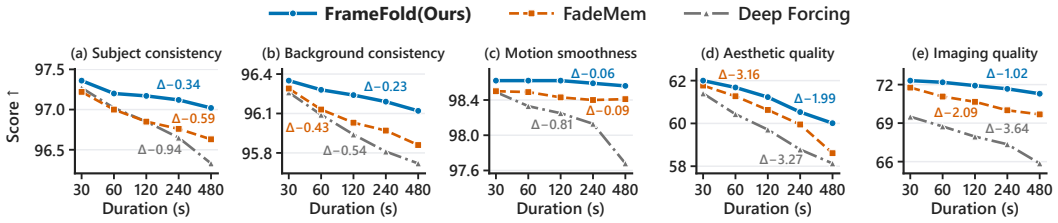

Figure 5: **Quantitative comparison of FrameFold with baselines on video generation up to 480 seconds.** We evaluate Self Forcing on 24 prompts with a cache budget of 12 frame-equivalent KV entries, using nested prefixes of continuous rollouts. Colored labels show $\Delta = s_{480s} - s_{30s}$; the duration axis uses a base-2 log scale.

Table 2: **Ablation on merge selection and value fitting.** Self Forcing on 48 prompts. K/V/O diff measure key/value/conditional-response differences; Span/Optim use span-weighted/fitted values. Best scores are in **bold**.

Selection	Value merging	Subject Consist.↑	Background Consist.↑	Motion Smooth.↑	Dynamic Degree↑	Aesthetic Quality↑	Imaging Quality↑
V diff	Span	97.31	96.26	98.17	**62.36**	60.33	70.58
	Optim	97.37	96.31	98.19	60.41	60.39	70.57
K diff	Span	97.40	96.25	98.17	61.11	60.34	70.56
	Optim	97.45	96.33	98.20	59.44	60.38	70.69
O diff	Span	97.48	96.33	98.25	57.64	**60.73**	70.67
	Optim	**97.66**	**96.43**	**98.28**	59.31	60.66	**70.80**

Table 3: **Ablation on the response objective.** Self Forcing on 128 prompts. Each response objective is used for both value fitting and merge selection. Best scores are in **bold**.

Response objective	Subject Consist.↑	Background Consist.↑	Motion Smooth.↑	Dynamic Degree↑	Aesthetic Quality↑	Imaging Quality↑
Full-cache	96.94	96.15	98.08	**61.56**	57.55	68.45
Conditional	**97.16**	**96.29**	**98.25**	60.42	**58.27**	**68.52**

1.14-point reduction in dynamic degree. These results support block-local response matching as a consolidation objective; qualitative comparisons appear in Appendix A.6.

Sensitivity. Appendix A.8 shows that FrameFold retains its quality advantage across cache budgets and maintains similar scores with moderate spatial pooling.

6 Conclusion

We presented FrameFold, a training-free KV cache consolidation method for autoregressive long-video generation under a fixed budget. By comparing conditional attention responses before and after merging, FrameFold guides both the construction of shared historical representatives and the selection of adjacent merges while preserving complete spatial grids. Experiments on Self Forcing, Causal Forcing, and LongLive demonstrate improved consistency and visual quality under matched cache budgets, with smaller quality declines in continuous eight-minute Self Forcing rollouts. Ablation studies further support conditional-response matching as a consolidation objective and the contributions of representative construction and merge selection.

AI USE STATEMENT

In this work, we used generative AI tools to assist with research ideation and technical discussion, retrieval and discovery of relevant literature and information, and editing the manuscript for clarity, grammar, and readability. AI-generated research suggestions were treated as preliminary input and were independently evaluated by the authors. Relevant literature and factual information suggested by AI tools were manually verified against the original sources. AI-assisted manuscript edits were reviewed to ensure that they preserved the intended technical meaning, experimental claims, and citations.

The proposed method, implementation, experimental results, and scientific conclusions were ultimately developed, verified, and validated by the authors. We did not use generative AI tools to generate synthetic datasets or to prove mathematical claims. We take responsibility for the final content of this work, including text, claims, code, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4 specifies the conditional-response objective, representative construction, and merge-selection procedure. Section 5.1 describes the experimental setup and evaluation metrics, while Appendix A provides study-specific protocols and additional results. Appendix B details the pooling scheme, batched candidate evaluation, and baseline configurations. Implementation code is not included with this submission. We plan to release the source code publicly, subject to the required internal approvals.

REFERENCES

Yuxuan Bian, Zeyue Xue, Songchun Zhang, Shiyi Zhang, Weiyang Jin, Yaowei Li, Junhao Zhuang, Haoran Li, Jie Huang, Haoyang Huang, Nan Duan, and Qiang Xu. Echo-Infinity: Learning evolving memory for real-time infinite video generation. *arXiv preprint arXiv:2606.04527*, 2026.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

Peiliang Cai, Evelyn Zhang, Jiacheng Liu, Hao Lin, Ruiqi Zhang, Weile Mo, Yue Ma, Shikang Zheng, Jiehang Huang, Dongrui Liu, and Linfeng Zhang. Focused Forcing: Content-aware per-frame KV selection for efficient autoregressive video diffusion. *arXiv preprint arXiv:2605.18346*, 2026.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Conference on Language Modeling*, 2025.

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*, 2024.

Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, Weiming Xiong, Wei Wang, Nuo Pang, Kang Kang, Zhiheng Xu, Yuzhe Jin, Yupeng Liang, Yubing Song, Peng Zhao, Boyuan Xu, Di Qiu, Debang Li, Zhengcong Fei, Yang Li, and Yahui Zhou. SkyReels-V2: Infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025.

Hanmo Chen, Chenghao Xu, Xu Yang, Xuan Chen, and Cheng Deng. Past- and future-informed KV cache policy with salience estimation in autoregressive video diffusion. *arXiv preprint arXiv:2601.21896*, 2026a.

Shuo Chen, Cong Wei, Sun Sun, Tiancheng Shen, Ping Nie, Kai Zou, Ge Zhang, Ming-Hsuan Yang, and Wenhu Chen. Context Forcing: Consistent autoregressive video generation with long context. In *Proceedings of the International Conference on Machine Learning*, 2026b.

Jiaxing Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-Forcing++: Towards minute-scale high-quality video generation. In *International Conference on Learning Representations*, 2026.

Zelin Gao, Qiuyu Wang, Jiapeng Zhu, Jingye Chen, Zichen Liu, Qingyan Bai, Jiahao Wang, Yufeng Yuan, Hanlin Wang, Yichong Lu, Ka Leong Cheng, Haojie Zhang, Jian Gao, Tianrui Feng, Yuzheng Liu, Yao Yao, Yinghao Xu, Xing Zhu, Yujun Shen, and Hao Ouyang. Infinite worlds with versatile interactions. *arXiv preprint arXiv:2607.07534*, 2026.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, 2024.

Hang Guo, Zhaoyang Jia, Jiahao Li, Bin Li, Yuanhao Cai, Jiangshan Wang, Yawei Li, and Yan Lu. Efficient autoregressive video diffusion with dummy head. *arXiv preprint arXiv:2601.20499*, 2026a.

Yuwei Guo, Ceyuan Yang, Hao He, Yang Zhao, Meng Wei, Zhenheng Yang, Weilin Huang, and Dahua Lin. End-to-end training for autoregressive video diffusion via self-resampling. In *Proceedings of the European Conference on Computer Vision*, 2026b.

Qixin Hu, Shuai Yang, Wei Huang, Song Han, and Yukang Chen. LongLive-RAG: A general retrieval-augmented framework for long video generation. *arXiv preprint arXiv:2606.02553*, 2026.

Siqiao Huang, Jialong Wu, Qixing Zhou, Shangchen Miao, and Mingsheng Long. Vid2World: Crafting video diffusion models to interactive world models. In *International Conference on Learning Representations*, 2026a.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, 2025.

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.

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

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.

Youngrae Kim, Qixin Hu, C.-C. Jay Kuo, and Peter Beerel. MemRoPE: Training-free infinite video generation via evolving memory tokens. In *Proceedings of the European Conference on Computer Vision*, 2026.

Haodong Li, Shaoteng Liu, Zhe Lin, and Manmohan Chandraker. Rolling Sink: Bridging limited-horizon training and open-ended testing in autoregressive video diffusion. *arXiv preprint arXiv:2602.07775*, 2026a.

Wuyang Li, Wentao Pan, Po-Chien Luan, Yang Gao, and Alexandre Alahi. Stable Video Infinity: Infinite-length video generation with error recycling. In *International Conference on Learning Representations*, 2026b.

Xirui Li, Chao Ma, Xiaokang Yang, and Ming-Hsuan Yang. VidToMe: Video token merging for zero-shot video editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024a.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024b.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems*, 2024.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling Forcing: Autoregressive long video diffusion in real time. In *International Conference on Learning Representations*, 2026.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems*, 2023.

Yu Lu, Junjie Yang, Piotr Koniusz, YuXin Song, and Yi Yang. FadeMem: Distance-aware memory consolidation for autoregressive video diffusion. *arXiv preprint arXiv:2606.10671*, 2026.

Jiayi Luo, Qiyan Liu, Tengyang Wang, JunHao Liu, Jiayu Chen, Cong Wang, Hanxin Zhu, Chen Gao, Xiaobin Hu, Qingyun Sun, and Zhibo Chen. Future Forcing: Future-aware training-free KV cache policy for autoregressive video generation. *arXiv preprint arXiv:2605.30083*, 2026.

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.

Xiaofeng Mao, Zhen Li, Chuanhao Li, Xiaojie Xu, Kaining Ying, Tong He, Jiangmiao Pang, Yu Qiao, and Kaipeng Zhang. Yume-1.5: A text-controlled interactive world generation model. *arXiv preprint arXiv:2512.22096*, 2025.

Yu Meng, Xiangyang Luo, Letian Li, Wenyuan Jiang, Chen Gao, Xinlei Chen, Yong Li, and Xiao-Ping Zhang. TetherCache: Stabilizing autoregressive long-form video generation with gated recall and trusted alignment. *arXiv preprint arXiv:2606.13035*, 2026.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie Gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Runjia Qian, Zile Wang, Jihai Zhang, Kai Zou, Wei Yu, Jiaxing Li, Zexiang Liu, Yaokun Li, Fei Kang, Kaichen Huang, Mengyin An, Haobo Zhang, Biao Jiang, Jiahua Wang, Haofeng Sun, Yang Liu, and Yangguang Li. Matrix-Game 3.5: Enhancing real-time streaming interactive world models with patch memory. *arXiv preprint arXiv:2608.29910*, 2026.

Dvir Samuel, Issar Tzachor, Matan Levy, Michael Green, Gal Chechik, and Rami Ben-Ari. FASTAR: Fast autoregressive video diffusion and world models with temporal cache compression and sparse attention. In *Proceedings of the International Conference on Machine Learning*, 2026.

Sand.ai, Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, W. Q. Zhang, Weifeng Luo, Xiaoyang Kang, Yuchen Sun, Yue Cao, Yunpeng Huang,

Yutong Lin, Yuxin Fang, Zewei Tao, Zheng Zhang, Zhongshu Wang, Zixun Liu, Dai Shi, Guoli Su, Hanwen Sun, Hong Pan, Jie Wang, Jiexin Sheng, Min Cui, Min Hu, Ming Yan, Shucheng Yin, Siran Zhang, Tingting Liu, Xianping Yin, Xiaoyu Yang, Xin Song, Xuan Hu, Yankai Zhang, and Yuqiao Li. MAGI-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

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. In *Proceedings of the International Conference on Machine Learning*, 2026.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive KV cache merging for LLMs on long-context tasks. *arXiv preprint arXiv:2407.08454*, 2024.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

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.

Jung Yi, Wooseok Jang, Paul Hyunbin Cho, Jisu Nam, Heeji Yoon, and Seungryong Kim. Deep Forcing: Training-free long video generation with deep sink and participative compression. In *Proceedings of the International Conference on Machine Learning*, 2026.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models. In *Advances in Neural Information Processing Systems*, 2025.

Lvmin Zhang, Shengqu Cai, Muyang Li, Chong Zeng, Beijia Lu, Anyi Rao, Song Han, Gordon Wetzstein, and Maneesh Agrawala. TinyHistory: Lightweight video history embeddings via two-stage context learning. In *Proceedings of the European Conference on Computer Vision*, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Wenqu Zhao, Xuemin Chi, Xin Zhang, Guoqing Ma, Baorun Li, Jianjie Fang, Peizhi Tang, Chen Gao, and Wei Wu. DensityKV: Density-guided KV cache compression for long video generation. *arXiv preprint arXiv:2608.27922*, 2026a.

Zengqun Zhao, Yanzuo Lu, Ziquan Liu, Jifei Song, Jiankang Deng, and Ioannis Patras. Relax Forcing: Relaxed KV-memory for consistent long video generation. In *Proceedings of the British Machine Vision Conference*, 2026b. Accepted.

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. In *Proceedings of the International Conference on Machine Learning*, 2026.

Tianrui Zhu, Shiyi Zhang, Zhirui Sun, Jingqi Tian, and Yansong Tang. Memorize-and-Generate: Towards long-term consistency in real-time video generation. *arXiv preprint arXiv:2512.18741*, 2025.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast KV compaction via Attention Matching. In *Proceedings of the International Conference on Machine Learning*, 2026.

APPENDIX CONTENTS

A Supplementary Experiments and Analyses **16**

A.1 Motivation Analysis: Spatial Attention and Token Selection 16

A.2 Motivation Analysis: Attention Coverage and Generation Quality 16

A.3 Supplementary Results: Qualitative Comparisons 17

A.4 Supplementary Results: Long-Horizon Generation 18

A.5 Supplementary Results: Merge Selection and Value Fitting 19

A.6 Supplementary Results: Response Objective 20

A.7 Additional Experiments: Evaluation on LongLive 21

A.8 Additional Experiments: Sensitivity to Cache Budget and Pooling 23

A.9 Additional Experiments: Generation Throughput 23

B Implementation Details **24**

B.1 Efficient Implementation of FrameFold 24

B.2 Baseline Configurations 25

C Discussion **26**

C.1 Limitations 26

A Supplementary Experiments and Analyses

A.1 Motivation Analysis: Spatial Attention and Token Selection

Analysis objective. Queries at different spatial locations share the same retained history, so a shared token selection must serve their different retrieval needs. We supplement the analysis of shared ranking in Section 3.1 through query-specific attention distributions and the spatial patterns produced by actual Top-K selection.

Query-specific attention patterns. Figure 6 shows attention distributions for queries at the upper-left, center, and lower-right locations in layers 0, 10, 20, and 29. Within the same layer, regions with high attention differ across queries: in layers 10 and 20, the hotspots are concentrated near the respective query locations. The distributions also vary across layers. For example, attention is more dispersed in layer 0, and the lower-right query also exhibits a broader response pattern in layer 29. These observations illustrate differences in the historical content attended to by spatial queries; the regions emphasized by one query do not represent the needs of the others.

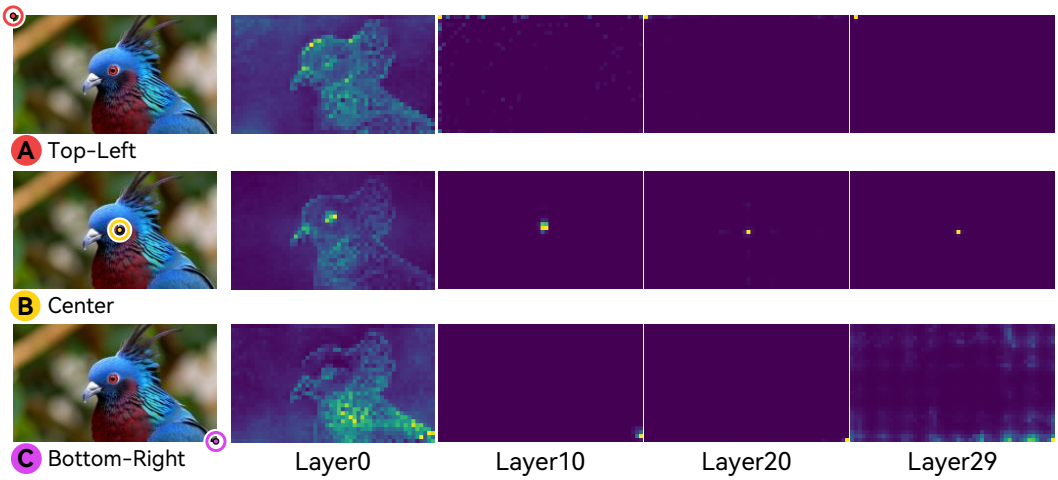

Figure 6: **Visualization of query-specific spatial attention patterns.** Each row shows a query at the upper-left, center, or lower-right spatial location, followed by its attention maps in Transformer layers 0, 10, 20, and 29.

Spatial selection patterns. We further examine actual Top-K selection in Deep Forcing. Figure 7 shows selection-frequency maps from seven layers. Layers 7, 11, 15, 18, and 22 exhibit scattered local concentrations and bands of high selection frequency. In layer 26, selection frequencies are higher on the left and right sides and lower in the center. Layer 29 exhibits two prominent vertical bands of high selection frequency, with a large central region close to zero. These maps illustrate the spatial retention patterns produced by shared Top-K ranking: selection frequencies are uneven across spatial locations, and the distributions vary across layers. Together with the query-level coverage gaps in Section 3.1, these observations motivate accounting for retrieval needs at different spatial locations during history compression and preserving complete spatial representations.

A.2 Motivation Analysis: Attention Coverage and Generation Quality

Experimental setup. This section supplements the Top-K/Bottom-K comparison in Section 3.1 with key experimental settings and temporal results. We generate 60-second videos on 48 matched prompts using Self Forcing and Causal Forcing. Both policies retain 50% of the candidate history tokens and share the same attention-scoring procedure, prompts, random seeds, and sampling settings; they retain the highest-ranked and lowest-ranked candidates, respectively. For the attention coverage in Figure 2(a), we measure the attention assigned by each query to retained tokens as a fraction of its total attention within the candidate history.

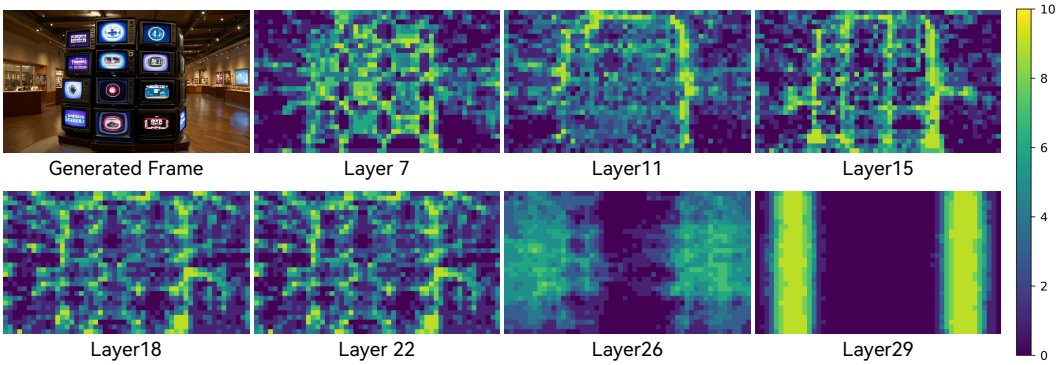

Figure 7: **Visualization of spatial selection patterns in Deep Forcing.** A generated frame is shown alongside selection-frequency maps for seven layers: 7, 11, 15, 18, 22, 26, and 29. Color indicates selection frequency at each spatial location.

Temporal results. Figure 8 extends the first- and final-frame comparison in Figure 2(c), showing results at 0, 15, 30, 45, and 60 seconds. In the Causal Forcing gallery example, the intermediate frames at 15–45 seconds under Top-K already exhibit lower overall brightness and changes in background color, followed by a pronounced blue cast in the final 60-second frame. Bottom-K better preserves the warm tones of the initial gallery background across the sampled timestamps. These intermediate frames show how appearance differences between the two retention policies develop over time, providing a visual complement to the observation in the main text that higher current attention coverage need not yield better subsequent generation quality.

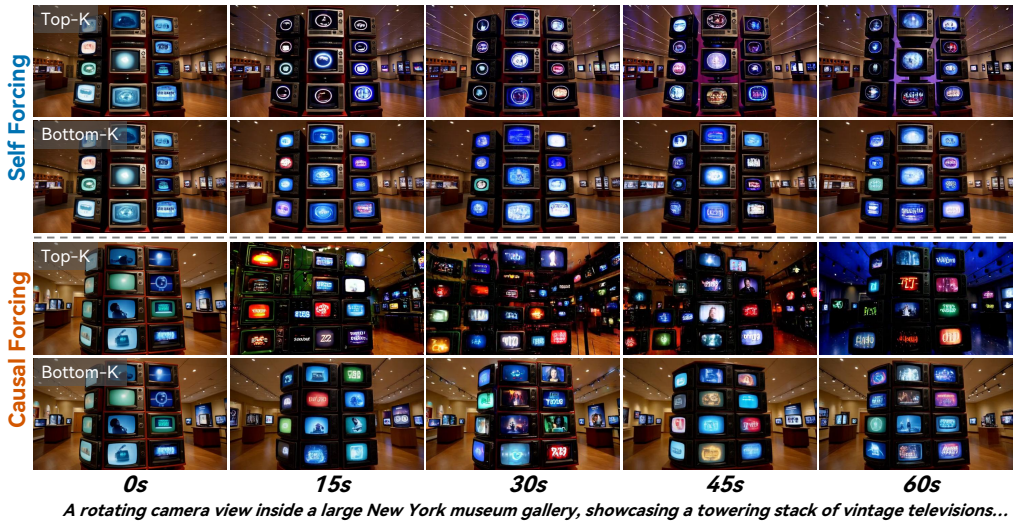

Figure 8: **Qualitative comparison of Top-K and Bottom-K over 60 seconds of generation.** Results are shown for Self Forcing and Causal Forcing. Both policies retain 50% of the candidate history using the same scoring procedure, with matched prompts, seeds, and sampling settings.

A.3 Supplementary Results: Qualitative Comparisons

Figure 9 supplements the main comparison in Section 5.2 with two additional examples under the same 60-second generation settings. The Self Forcing example shows a panda playing a guitar, and the Causal Forcing example shows milk being poured into a cup.

In the panda example, the original Self Forcing model develops colored stripe artifacts in the bamboo background, while FadeMem exhibits changes in the panda's facial appearance and background

color. FrameFold maintains a more consistent panda appearance and more stable background colors. In the milk-pouring example, the original Causal Forcing model and Deep Forcing exhibit vertical stripe artifacts in later frames, whereas FrameFold maintains cleaner cup contours and background textures.

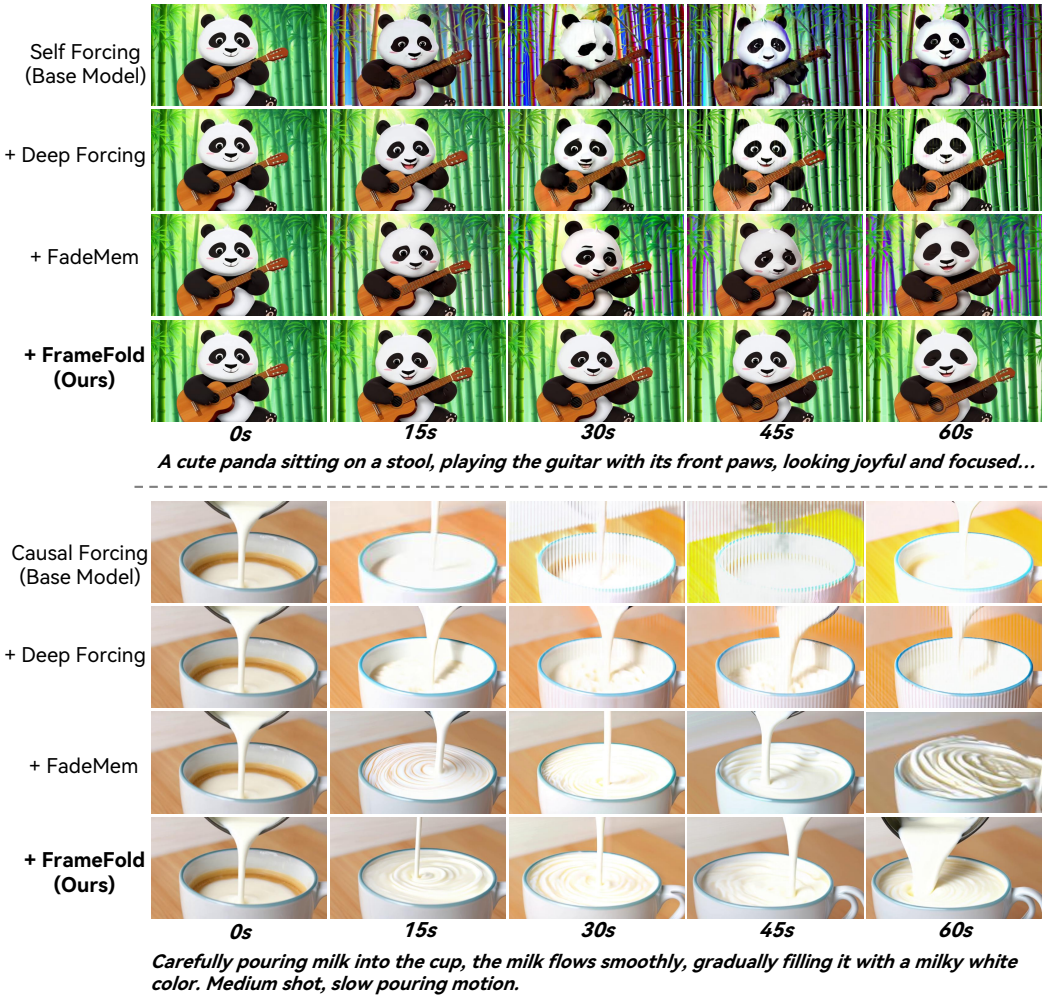

Figure 9: **Additional qualitative comparisons on 60-second video generation.** Self Forcing and Causal Forcing examples are shown in the top and bottom groups, respectively, with frames sampled at 0, 15, 30, 45, and 60 seconds. Methods within each example use the same prompt, seed, and sampling settings, with a cache budget of 12 frame-equivalent KV entries.

A.4 Supplementary Results: Long-Horizon Generation

Evaluation protocol. This section supplements the long-horizon experiment in Section 5.3 with complete results across all six VBench-Long dimensions and temporal visualizations. We compare FrameFold, FadeMem, and Deep Forcing using Self Forcing on 24 prompts, with a fixed cache budget of 12 frame-equivalent KV entries. Each method generates a continuous 480-second video for each prompt. We evaluate its nested prefixes at 30, 60, 120, 240, and 480 seconds to track generation performance as the evaluated duration increases.

Quantitative results. Table 4 shows that FrameFold achieves the highest subject consistency, background consistency, motion smoothness, aesthetic quality, and imaging quality among the three methods at every evaluated duration, with smaller declines from 30 to 480 seconds. For example, imaging quality decreases by 1.02 points for FrameFold, compared with 2.09 points for FadeMem

and 3.64 points for Deep Forcing. Over the same interval, FrameFold's motion smoothness changes from 98.62 to 98.56, while its dynamic degree increases from 67.45 to 69.79. Its consistency and visual-quality advantages thus accompany high motion smoothness and dynamic degree.

Table 4: **Quantitative comparison of FrameFold with baselines on video generation up to 480 seconds.** We evaluate Self Forcing on 24 prompts with a cache budget of 12 frame-equivalent KV entries. At each duration, we evaluate a nested prefix of the same continuous 480-second rollout. Higher is better; best scores within each duration group are in **bold**.

Duration	Method	Subject Consist.↑	Background Consist.↑	Motion Smooth.↑	Dynamic Degree↑	Aesthetic Quality↑	Imaging Quality↑
30s	Deep Forcing	97.27	96.26	98.49	55.99	61.40	69.50
	FadeMem	97.22	96.29	98.50	67.44	61.77	71.77
	FrameFold	**97.36**	**96.35**	**98.62**	**67.45**	**62.00**	**72.31**
60s	Deep Forcing	97.02	96.09	98.33	54.97	60.43	68.74
	FadeMem	97.00	96.13	98.49	**70.29**	61.27	71.07
	FrameFold	**97.20**	**96.28**	**98.62**	68.68	**61.68**	**72.18**
120s	Deep Forcing	96.85	95.94	98.25	50.96	59.71	67.95
	FadeMem	96.85	96.03	98.43	67.96	60.63	70.66
	FrameFold	**97.17**	**96.24**	**98.62**	**68.85**	**61.23**	**71.92**
240s	Deep Forcing	96.65	95.81	98.13	46.66	58.78	67.35
	FadeMem	96.76	95.97	98.40	67.42	59.95	70.00
	FrameFold	**97.12**	**96.19**	**98.59**	**68.73**	**60.53**	**71.67**
480s	Deep Forcing	96.33	95.72	97.68	41.93	58.13	65.86
	FadeMem	96.63	95.86	98.41	67.88	58.61	69.68
	FrameFold	**97.02**	**96.12**	**98.56**	**69.79**	**60.01**	**71.29**

Qualitative results. Figure 10 compares subject and scene evolution under FrameFold and FadeMem over eight minutes of generation, using a garden scene inside a glass sphere. In the displayed frames, FadeMem introduces additional figures at 200–250 seconds, followed by further changes in subject appearance and surrounding structures relative to the initial scene. FrameFold preserves a single central figure and a more stable overall layout. The raking tool's position and the sand patterns change over time, while the central figure remains recognizable in the final 480-second frame. This sequence provides a visual counterpart to the subject and background consistency results.

A.5 Supplementary Results: Merge Selection and Value Fitting

Experimental setup. This section supplements the ablation study in Section 5.4 with configuration details and qualitative comparisons. We combine three merge-selection criteria—K diff, V diff, and O diff—with two value-merging rules, Span and Optim, yielding six configurations. All six use Self Forcing to generate 30-second videos on 48 prompts, with a cache budget of 12 frame-equivalent KV entries, a pooling factor of 4, and queries from the three most recently completed frames for evaluation. All other generation settings remain fixed.

Ablation variants. K diff and V diff rank adjacent pairs by the mean L_2 distance between their pooled key or value features at corresponding spatial locations. O diff instead measures the conditional-response distortion between the original pair and its merged representative under the same recent queries. Each criterion selects the eligible pair with the lowest score. Keys are merged using temporal-span weights in every variant. Span applies these same weights to values, whereas Optim holds the merged keys fixed and fits a shared value-merging coefficient through the response-matching objective in Equation 6. This fitting objective is identical across all three selection criteria: for example, K diff + Optim uses key-feature distance for pair selection and conditional-response matching for value fitting. Pooling is used for candidate evaluation and coefficient fitting, while the selected merge is applied to the complete spatial grids. O diff + Optim corresponds to the full FrameFold configuration.



Figure 10: **Qualitative comparison of FrameFold and FadeMem over 480 seconds of generation.** FadeMem (top) and FrameFold (bottom) use Self Forcing with the same prompt and a cache budget of 12 frame-equivalent KV entries.

Qualitative results. Figure 11 shows the six configurations starting from the same initial frame. With Optim fixed, the selection criteria yield different subject depictions: at 10 seconds, V diff primarily shows a close-up of the white Samoyed, whereas O diff shows both the Samoyed and the golden retriever. With O diff fixed, the two dogs overlap substantially in the 10-second frame under Span, whereas Optim shows them separately in the foreground and background. Under O diff + Optim, the white Samoyed and golden retriever remain visually distinguishable in subsequent sampled frames, including the final 30-second frame. These paired comparisons illustrate differences in subject visibility and composition associated with merge selection and value construction, providing a visual complement to the quantitative ablation in Table 2.

A.6 Supplementary Results: Response Objective

Experimental setup. We compare conditional and full-cache response matching for both value fitting and merge selection, using Self Forcing to generate 30-second videos on 128 prompts with a cache budget of 12 frame-equivalent KV entries. Conditional matching compares the block-local responses of the original pair and its representative. Full-cache matching compares the attention outputs of the entire cache before and after the candidate replacement. All other generation and merge settings remain fixed.



Figure 11: **Ablation on merge selection and value fitting.** Qualitative results are shown for Self Forcing with 30-second videos and a cache budget of 12 frame-equivalent KV entries. O diff + Optim corresponds to FrameFold.

Qualitative results. Figure 12 shows a rider crossing a desert at sunset, complementing the quantitative results in Table 3. Both variants preserve the rider and horse across the sampled frames, with differences in the sky's color bands and the shape of the dust trail at 20 and 30 seconds.



Figure 12: **Ablation on the response objective.** Qualitative results show 30-second Self Forcing generation with a cache budget of 12 frame-equivalent KV entries. Full-cache (top) and conditional (bottom, FrameFold) use the same prompt; each objective is applied to both value fitting and merge selection.

A.7 Additional Experiments: Evaluation on LongLive

Experimental setup. To evaluate FrameFold on an additional backbone, we generate 120-second videos with LongLive (Yang et al., 2026) using the same 128 prompts as the main comparison.

Table 5: **Quantitative comparison on LongLive for 120-second video generation.** All methods use the same 128 prompts as the main comparison and a cache budget of 12 frame-equivalent KV entries. We report six VBench-Long dimensions; higher is better, and the best scores are in **bold**.

Config	Method	Subject Consist.↑	Bg. Consist.↑	Motion Smooth.↑	Dynamic Degree↑	Aesthetic Quality↑	Imaging Quality↑
120s	LongLive	97.24	96.22	98.43	**47.38**	57.61	66.93
	Deep Forcing	97.58	96.25	**98.53**	39.59	57.88	67.35
	FadeMem	97.45	96.39	98.47	45.96	58.53	67.65
	FrameFold	**97.76**	**96.75**	98.50	46.32	**59.09**	**68.10**

We compare the original LongLive model, Deep Forcing, FadeMem, and FrameFold using the six VBench-Long dimensions. All methods share a cache budget of 12 frame-equivalent KV entries, random seeds, and sampling settings. FrameFold uses a pooling factor of 4 and queries from the three most recently completed historical frames; other settings follow Section 5.1.

Quantitative results. Table 5 shows that FrameFold achieves the highest subject consistency, background consistency, aesthetic quality, and imaging quality among the compared methods. Background consistency and aesthetic quality exceed those of the strongest baseline, FadeMem, by 0.36 and 0.56 points, respectively. Motion smoothness is slightly lower than Deep Forcing's, while dynamic degree is lower than the original LongLive model's and higher than both Deep Forcing's and FadeMem's. These results extend the consistency and visual-quality benefits of FrameFold to LongLive at 120 seconds.

Qualitative results. Figure 13 shows a small figure raking a Zen garden inside a glass sphere. In the original LongLive and FadeMem sequences, the initial figure changes into larger animal-like characters, while Deep Forcing introduces multiple figures in later frames. FrameFold retains a single small humanoid figure across the sampled timestamps, with changes in the rake position and sand patterns. This example illustrates the subject-preservation differences reflected in the quantitative comparison.

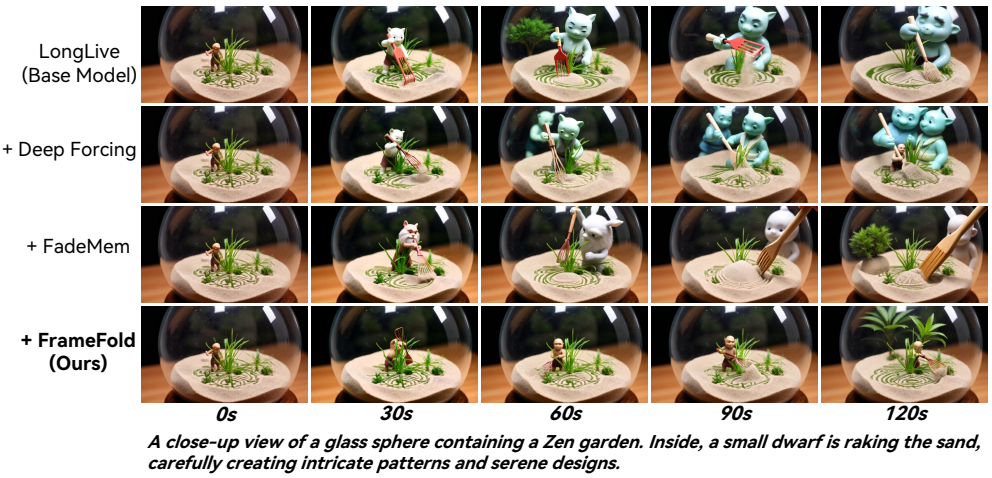

Figure 13: **Qualitative comparison on LongLive over 120 seconds of generation.** From top to bottom: the original LongLive model, Deep Forcing, FadeMem, and FrameFold. Frames are shown at 0, 30, 60, 90, and 120 seconds. All methods use the same prompt and random seed, with a cache budget of 12 frame-equivalent KV entries.

A.8 Additional Experiments: Sensitivity to Cache Budget and Pooling

Experimental setup. This section examines how the cache budget and spatial pooling factor affect the generation quality of FrameFold. The cache budget controls the number of stored frame-equivalent KV entries, while the pooling factor controls the spatial resolution of the proxies used for value fitting and merge scoring. Both studies use Self Forcing to generate 60-second videos on 48 prompts. The cache-budget study compares 12, 15, 18, and 21 entries with the pooling factor fixed at 4. The spatial-pooling study compares factors of 1, 2, 4, and 8 with the cache budget fixed at 12 entries; a factor of 1 uses the original spatial resolution.

Cache budget. Table 6 shows that FrameFold outperforms Deep Forcing in subject consistency, background consistency, motion smoothness, aesthetic quality, and imaging quality at every tested budget, with imaging-quality gains of 2.05–3.42 points. Using 12 entries, FrameFold also exceeds Deep Forcing with 21 entries across all six metrics: it uses 42.9% fewer cache entries while improving aesthetic quality and imaging quality by 2.11 and 3.66 points, respectively. For FrameFold, increasing the budget does not yield monotonic improvements across the metrics, and different metrics reach their highest scores at different budgets. Since a consolidated entry can represent multiple historical frames, the cache budget determines the granularity of the history representation, and the number of entries does not directly specify the historical span they cover. These results show that FrameFold maintains its quality advantage across the tested budgets and support using 12 entries as a compact default configuration.

Table 6: **Ablation on KV cache budget.** Experiments on Self Forcing use 48 prompts and 60-second videos. FrameFold uses a pooling factor of 4. Cache budgets are measured in frame-equivalent KV entries; higher scores are better. The default FrameFold configuration is highlighted in blue.

Method	Cache Budget	Subject Consist.↑	Background Consist.↑	Motion Smooth.↑	Dynamic Degree↑	Aesthetic Quality↑	Imaging Quality↑
Deep Forcing	12	96.75	95.98	98.09	61.32	59.63	68.85
	15	96.65	95.73	98.02	66.32	58.90	68.34
	18	96.64	95.83	97.98	61.74	58.88	68.11
	21	96.47	95.65	97.96	59.37	58.44	67.24
FrameFold	12	97.15	96.27	98.27	67.08	60.55	70.90
	15	97.19	96.21	98.28	64.37	60.28	70.74
	18	96.93	96.08	98.25	68.26	60.03	70.58
	21	96.94	96.09	98.41	64.58	59.88	70.66

Spatial pooling. Spatial pooling constructs the proxies used for value fitting and merge scoring, and the resulting merge coefficients are applied to the complete spatial KV grids. Across the four pooling settings, subject consistency, background consistency, and motion smoothness each vary by at most 0.16 points (Table 7). The default factor of 4 achieves the highest background consistency and aesthetic quality, with imaging quality of 70.90, close to the 70.97 obtained at factor 1. Factor 2 achieves the highest imaging quality of 71.30, while factor 8 reduces it to 70.37. At factor 4, each historical entry's original 30×52 grid is reduced to an 8×13 proxy grid, decreasing the token count from 1,560 to 104, while the stored KV representatives retain their full spatial resolution. Factor 4 thus maintains similar generation quality with substantially smaller evaluation proxies, supporting its use as the default pooling configuration.

A.9 Additional Experiments: Generation Throughput

Experimental setup. We measure generation throughput in output frames per second (FPS) using Self Forcing on NVIDIA H20 hardware, excluding VAE decoding and video saving from the timing. All four methods use a cache budget of 12 frame-equivalent KV entries, and we additionally evaluate FIFO and Deep Forcing with 21 entries. FrameFold uses a pooling factor of 4 and queries from the

Table 7: **Ablation on spatial pooling factor.** Experiments on Self Forcing use 48 prompts, 60-second videos, and a cache budget of 12 frame-equivalent KV entries. A pooling factor of 1 uses the original spatial resolution. Higher scores are better. The default configuration is highlighted in blue.

Backbone	Pooling Factor	Subject Consist.↑	Background Consist.↑	Motion Smooth.↑	Dynamic Degree↑	Aesthetic Quality↑	Imaging Quality↑
Self Forcing	1	97.12	96.19	98.29	66.39	60.29	70.97
	2	97.20	96.20	98.28	65.69	60.24	71.30
	4	97.15	96.27	98.27	67.08	60.55	70.90
	8	97.09	96.11	98.17	67.01	60.28	70.37

three most recently completed historical frames for value fitting and merge scoring. Other settings follow the defaults in Section 5.1.

Results and analysis. At the same budget of 12 entries, FrameFold achieves 5.56 FPS, compared with 6.39 for FIFO, 6.21 for FadeMem, and 6.18 for Deep Forcing (Figure 14). Its throughput is 13.0%, 10.5%, and 10.0% lower than that of these three methods, respectively. Together with the quality results at the same budget in Table 1, these measurements characterize the quality–throughput trade-off of FrameFold: it achieves higher consistency and visual quality within a fixed cache capacity while requiring more generation time. As an additional comparison across cache configurations, FrameFold with 12 entries achieves $1.31\times$ the throughput of Deep Forcing with 21 entries and obtains higher scores across all six dimensions in the cache-budget study (Table 6).

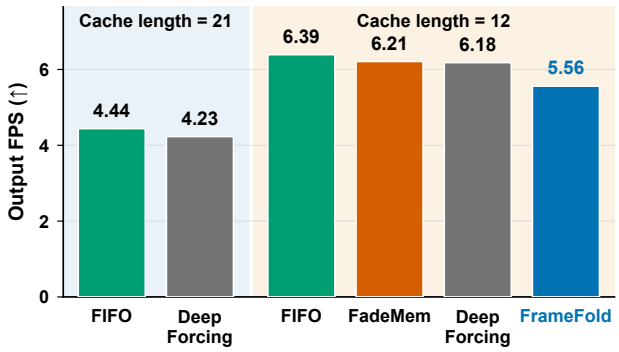

Figure 14: **Quantitative comparison of generation throughput.** Throughput is measured in output frames per second (FPS) on Self Forcing with an NVIDIA H20, excluding VAE decoding and video saving. All four methods use a cache budget of 12 frame-equivalent KV entries; FIFO and Deep Forcing are additionally evaluated with 21 entries.

B Implementation Details

B.1 Efficient Implementation of FrameFold

Pooled representations. We use adaptive average pooling to construct compact representations for fitting and scoring. Keys and values are spatially pooled independently within each historical entry, while queries from the latest R completed historical frames are jointly pooled over time and space into a single set of query probes. Our default uses $P = 4$ and $R = 3$, reducing the 30×52 spatial grid to 8×13. This reduces each historical entry from 1,560 tokens to 104 tokens and produces 104 query probes. Fitting and scoring use these pooled representations, and the selected mixing coefficients are applied to the complete spatial grids.

Key merging and position conventions. We merge keys before RoPE and add the two entries' temporal spans. Starting from unit spans, repeated span-weighted merging keeps each representative's keys equal to the mean of the original frame keys it covers. For response evaluation, the original keys use their respective current cache positions, while the merged keys use the left entry's position. Queries retain the positional encoding from the start of the cache update. Queries and keys receive RoPE before pooling; values are pooled directly.

Degenerate value fitting. When $D_i = 0$ in Section 4.3, the representative's response is independent of α. We set $\alpha_i^\star = 0$, which is a minimizer of the squared response error in this case.

Batched candidate evaluation. At each layer and merge iteration, we organize all adjacent candidate pairs along a candidate dimension and share the same query probes across them. Batched tensor operations compute conditional responses, fit value-mixing coefficients, and score response distortions for all candidates. Scoring reuses intermediate results from fitting to directly obtain the fitted representative responses. Each iteration selects and executes one merge, then evaluates the updated candidate set in the next batch of computations. Query probes remain fixed throughout the cache update.

Algorithm 1 summarizes one cache update, performed independently at each layer. Each entry stores (K^{pre}, V, s), and the budget includes the current chunk. The budget must accommodate all protected entries and at least one representative of the candidate history.

Algorithm 1 FrameFold cache consolidation for one layer

Require: Ordered cache $\mathcal{H}$ including the current chunk, total budget B, recent historical queries Q encoded at the start of the update, pooling factor P
Ensure: Updated cache $\mathcal{H}$ with $|\mathcal{H}| \le B$

Fix the protected sink, recent-history, and current-chunk entries
$\widehat{Q} \leftarrow \mathrm{PoolQueries}(Q, P)$
while $|\mathcal{H}| > B$ **do**
  $\mathcal{E} \leftarrow$ left-entry indices of adjacent pairs in the unprotected region
  Rebuild pooled KV proxies at the entries' current cache positions p_j
  for all $i \in \mathcal{E}$ **in parallel do**
    $\beta_i \leftarrow s_i/(s_i + s_{i+1})$
    $\widehat{\widehat{K}}_i \leftarrow \mathrm{Pool}_P\big(\mathrm{RoPE}_{p_i}(\beta_i K_i^{\mathrm{pre}} + (1-\beta_i)K_{i+1}^{\mathrm{pre}})\big)$
    Compute $\widehat{O}_i^-$ and A_i^+ under $\widehat{Q}$ using block-local attention
    $C_i \leftarrow A_i^+ \widehat{V}_{i+1}$; $D_i \leftarrow A_i^+(\widehat{V}_i - \widehat{V}_{i+1})$
    Fit $\alpha_i^\star$ using Equation 7; set $\alpha_i^\star = 0$ if $D_i = 0$
    $\widehat{O}_i^\star \leftarrow C_i + \alpha_i^\star D_i$
    $c_i \leftarrow (n_h U)^{-1} \sum_{h,u} \|\widehat{O}_{i,hu}^- - \widehat{O}_{i,hu}^\star\|_2$
  end for
  $i^\star \leftarrow \arg\min_{i \in \mathcal{E}} c_i$
  $\widetilde{K}^{\mathrm{pre}} \leftarrow \beta_{i^\star} K_{i^\star}^{\mathrm{pre}} + (1-\beta_{i^\star})K_{i^\star+1}^{\mathrm{pre}}$
  $\widetilde{V} \leftarrow \alpha_{i^\star}^\star V_{i^\star} + (1-\alpha_{i^\star}^\star)V_{i^\star+1}$
  Replace entries $i^\star, i^\star+1$ by $(\widetilde{K}^{\mathrm{pre}}, \widetilde{V}, s_{i^\star} + s_{i^\star+1})$
  Update cache positions
end while
return $\mathcal{H}$

B.2 Baseline Configurations

Shared cache conventions. All baselines follow the cache-budget convention in Section 5.1: the default budget is 12 frame-equivalent KV entries, including the three-entry current chunk, with three initial sink frames and the three most recent historical frames protected. Other budgets are specified in the corresponding experiments. Within each backbone, all methods use the same random seed and sampling settings for each prompt, ensuring identical generated content before the first cache eviction or merge and isolating the effect of the KV cache management policy on generation results.

Original backbones. Self Forcing and Causal Forcing use their original FIFO cache eviction policies and are listed under their model names in Table 1.

Deep Forcing. We use the official Deep Forcing implementation (Yi et al., 2026). Each eviction event removes a number of tokens equivalent to three complete spatial grids.

FadeMem. We implement FadeMem's inference-time memory policy (Lu et al., 2026) in the MemRoPE framework (Kim et al., 2026), following the temporal allocation and merge rules in arXiv version 1 (v1) of FadeMem. We set the temporal allocation exponent to $\beta = 0.3$ and merge both keys and values using temporal-span weights.

C Discussion

C.1 Limitations

Future query demands. FrameFold fits and evaluates representatives using recent historical queries, without explicitly modeling changes in future query demands. Query-proxy construction and conditional-response matching can be designed separately. Future Forcing (Luo et al., 2026) constructs future query proxies from historical statistics, offering a direction for extending our evaluation probes.

Additional computational cost. Fitting and scoring candidate representatives introduce additional computation. Pooling, closed-form fitting, and batched evaluation reduce this cost, but the current implementation has lower throughput than the compared methods under the same cache budget (Appendix A.9). Latency-sensitive applications therefore require balancing gains in consistency and visual quality against generation speed. Improving the execution efficiency of candidate evaluation is a direction for future optimization.