IN-DISTRIBUTION FORCING FOR LONG VIDEO GENERATION AT TEST TIME

Anonymous authors
Paper under double-blind review

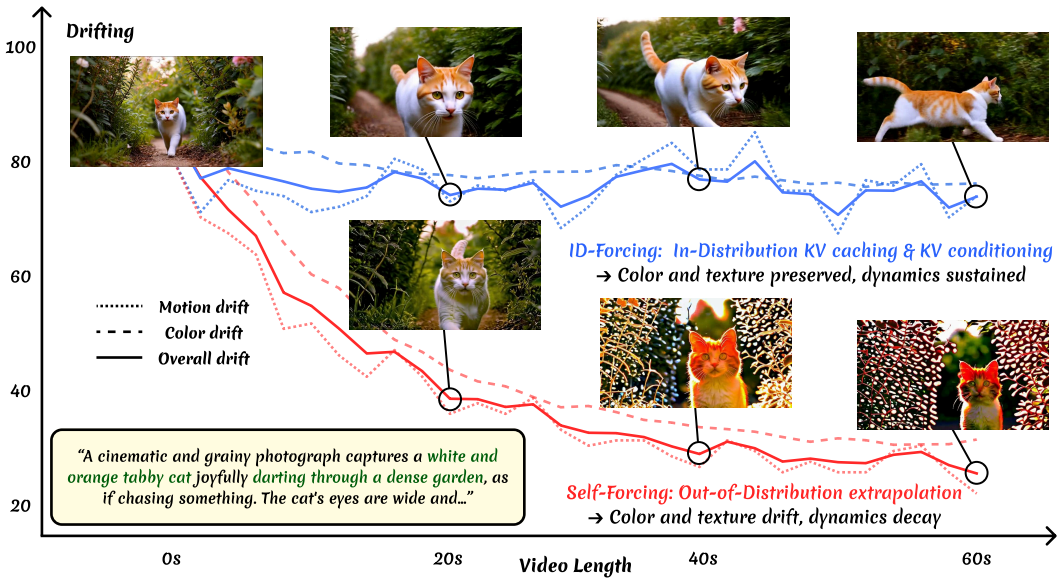


Figure 1: **Our ID-Forcing extends autoregressive long video generation far beyond the training horizon.** While vanilla extrapolation on a 5-second baseline (Self-Forcing) suffers severe quality decay, ID-Forcing maintains in-distribution KV operations to consistently suppress color and motion drift, achieving stable, minute-scale generation.

ABSTRACT

While modern autoregressive (AR) video diffusion models have demonstrated remarkable success in short-horizon video generation, generating long videos remains challenging due to a phenomenon known as *drifting*, where colors and textures shift, and motion dynamics decay. Existing approaches primarily rely on *KV conditioning*, which selects or modifies cached key-value (KV) entries to mitigate drifting. However, we observe that KV conditioning alone is insufficient because it assumes cached KV entries remain in-distribution. This assumption fails beyond the training horizon because no mechanism constrains the construction of cached KV entries during rollout, giving rise to the *KV-provenance problem* where cached entries themselves become out-of-distribution. To address this limitation, we propose **In-Distribution Forcing (ID-Forcing)**, a test-time framework that aligns both KV caching and KV conditioning with training configurations. Its key mechanism, *self-caching*, prevents out-of-distribution KV entries at their source: each chunk is cached without attending to any prior KV entry, keeping the rolling window exactly in-distribution without retraining. Consequently, ID-Forcing seamlessly extends short-horizon models to minute-scale long video generation. Extensive evaluations show that our method remains competitive on standard video generation benchmarks while substantially outperforming prior work in mitigating drifting, as validated by both our drift metrics and a user study. Video results are available at https://idforcing.github.io.

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.

1 INTRODUCTION

Recent advances in video diffusion models (Google DeepMind, 2025; Kling Team, Kuaishou Technology, 2025; OpenAI, 2025; Wan et al., 2025) enable high-fidelity synthesis of short video clips with coherent motion, yet interactive applications such as world models (Hong et al., 2025; Shen et al., 2026; Huang et al., 2025a), streaming content (Kodaira et al., 2026; Henschel et al., 2025; Chen et al., 2025), and long-form storytelling (Elmoghany et al., 2026; Luo et al., 2026) demand generation far beyond a few seconds. To enable such long-horizon generation, autoregressive (AR) video diffusion (Yin et al., 2025; Chen et al., 2024; Huang et al., 2025b; Yang et al., 2026) has emerged as the standard paradigm, generating videos sequentially in chunks while conditioning each new chunk on the key–value (KV) cache of previously generated content. Specifically, Self-Forcing (Huang et al., 2025b) reduces the training-inference gap by training on autoregressive rollouts, while LongLive (Yang et al., 2026) extends generation to longer horizons through streaming long-video tuning.

However, the ability to generate over extended horizons does not ensure that visual quality persists. As generation proceeds, colors and textures tend to shift, and motion dynamics gradually decay. This progressive degradation, known as *drifting*, arises because the rollout at inference differs from the rollout seen during training. Training on self-rollouts (Huang et al., 2025b; Cui et al., 2026; Yang et al., 2026; Chen et al., 2026; Liu et al., 2026) addresses this by exposing the model to its own generated context, but only within a finite training horizon. Extrapolating beyond this horizon therefore introduces conditioning configurations outside the training distribution, causing errors to accumulate over time. Yet simply extending the training horizon cannot solve this issue, as extrapolating past the horizon is inherently unavoidable in arbitrarily long video generation.

To address drifting, a large body of work controls what the model reads from the KV cache during generation, which we refer to as *KV conditioning*. Existing approaches typically select cached entries at the chunk level, as with attention sinks (Yi et al., 2026), or modify their temporal positional embeddings (Yesiltepe et al., 2026; Kim et al., 2026). These methods, however, are valid only under the assumption that the cached KV entries are in-distribution, as they consider only which entries to attend to and where to position them. This assumption has not been explicitly examined, and nothing ensures that it holds. Thus, understanding drifting requires examining not only the KV entries read during generation, but also the context under which they were constructed.

We observe that whether a KV entry is in-distribution is determined by *KV caching*, the operation that computes and stores a KV entry for each generated chunk. A KV entry depends not only on its own chunk but also on the other KV entries it attends to during KV caching, *i.e.,* its *provenance*. Beyond the training horizon, every chunk is cached under a provenance never seen during training, so the resulting KV entry is out-of-distribution. We term this the *KV-provenance problem* (Fig. 2). Since provenance is fixed at KV caching, no KV conditioning policy can recover from it, and keeping extrapolation in-distribution requires controlling *both KV caching and KV conditioning*.

Motivated by this insight, we propose **In-Distribution Forcing (ID-Forcing)**, a test-time method that aligns both KV caching and KV conditioning with the KV context configurations encountered during training. For KV caching, we introduce *self-caching*. Beyond the training horizon, the earlier KV entries are cached attending only to themselves, while the later ones are cached autoregressively on top of them, which prevents any unseen provenance from being constructed (§3.1). For KV conditioning, we show that the first chunk must remain in the conditioning window for it to stay in-distribution, and that self-caching keeps the rolling window exactly in-distribution (§3.2). Every KV operation is thus one that training executed. As a result, our method yields a remarkably simple framework: a plain rolling window with one rule for each KV operation, extending a five-second model to minute-scale generation.

Our contributions are as follows:

- We characterize the **KV-provenance problem** as a source of drift in long-video extrapolation and establish the importance of keeping both KV caching and KV conditioning within the configurations encountered during training (§3.1).
- We propose **In-Distribution Forcing (ID-Forcing)**, which introduces **self-caching** to control cache provenance and jointly aligns KV caching and KV conditioning with training (§3.1, §3.2).
- We demonstrate effective suppression of drift in minute-scale generation, preserving visual quality and temporal consistency while maintaining active motion (§4).

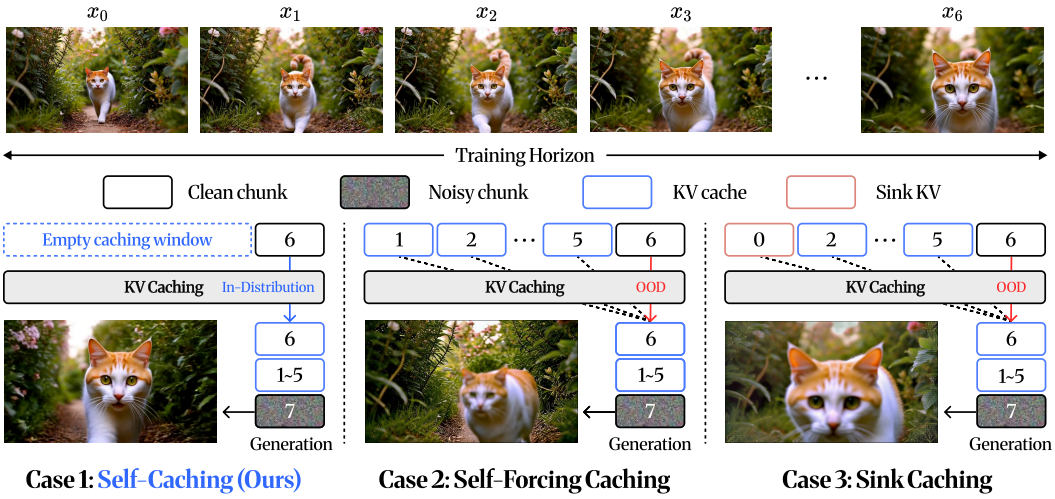


Figure 2: **Same conditioning, different provenance.** We generate the next chunk x_7 from the same conditioning window $\kappa_{1:6}$, varying only how the last chunk of the horizon, x_6, is cached into κ_6. **Case 1** (self-caching): x_6 attends to no prior entry, and x_7 shows no drifting. **Case 2** (self-forcing caching): x_6 attends to a window never seen in training, and x_7 degrades. **Case 3** (sink caching): keeping the first chunk does not help, as the window is still unseen in training. Thus, how a chunk is cached alone can determine whether the next chunk drifts.

2 PRELIMINARY

Autoregressive Video Diffusion. A latent video is a sequence of chunks $x_j \in \mathbb{R}^{f\times h\times w\times d}$, each covering f latent frames, generated under a text condition c. Chunked autoregressive (AR) diffusion models (Yin et al., 2025; Chen et al., 2024; Huang et al., 2025b) implement the factorization

$$p(x_{0:M} \mid c) \;=\; \prod_{j=0}^{M} p(x_j \mid x_{<j},\, c). \tag{1}$$

The conditioning on $x_{<j}$ is not implemented on raw latents: each chunk is written into a single key–value (KV) entry once generated, and later chunks are conditioned on these KV entries through the attention operation. We omit the shared condition c below.

KV Operations during Training. AR factorization in Eq. (1) is implemented by performing self-rollouts over a fixed horizon of N chunks ($i = 0, \ldots, N-1$) using a few-step denoiser G_θ with T denoising steps (Huang et al., 2025b). Here, we write x_i^t for chunk i at noise level $t \in \{0, \ldots, T\}$, where x_i^T is pure noise and $x_i^0 = x_i$ is clean. In this process, each step relies on two core KV operations: *KV conditioning* for x_i generation and *KV caching* for storing the KV states of x_i to generate the next chunk.

KV conditioning. To generate chunk x_i, the denoiser G_θ performs T denoising steps while attending to all preceding KV entries $\kappa_{<i}$:

$$x_i^{t-1} \;=\; G_\theta(x_i^t,\, t\,;\, \kappa_{<i}), \qquad t = T, \ldots, 1, \quad x_i^T \sim \mathcal{N}(0, I). \tag{2}$$

KV caching. After generating the clean chunk x_i, the same denoiser at $t = 0$ computes its corresponding KV entry κ_i by attending to the same KV set $\kappa_{<i}$:

$$\kappa_i \;=\; E(x_i^0;\, \kappa_{<i}) \;=\; \mathrm{KV}\Big[\,\overline{G}_\theta(x_i^0,\, t{=}0\,;\, \kappa_{<i})\Big]. \tag{3}$$

Here, $\mathrm{KV}[\cdot]$ extracts the KV states from the forward pass while discarding the denoising prediction, and the bar denotes a stop-gradient operator. For $i = 0$, the initial chunk attends solely to itself. As shown in Eqs. (2) and (3), the conditioning window and the caching window are the same set $\kappa_{<i}$, which grows from 0 to $N-2$ during training.

Drift during Extrapolation. Beyond the training horizon $i \geq N$, standard rolling-window methods fix both the window sizes to $L < N$, evicting the oldest entry κ_{i-L} when generating each chunk.

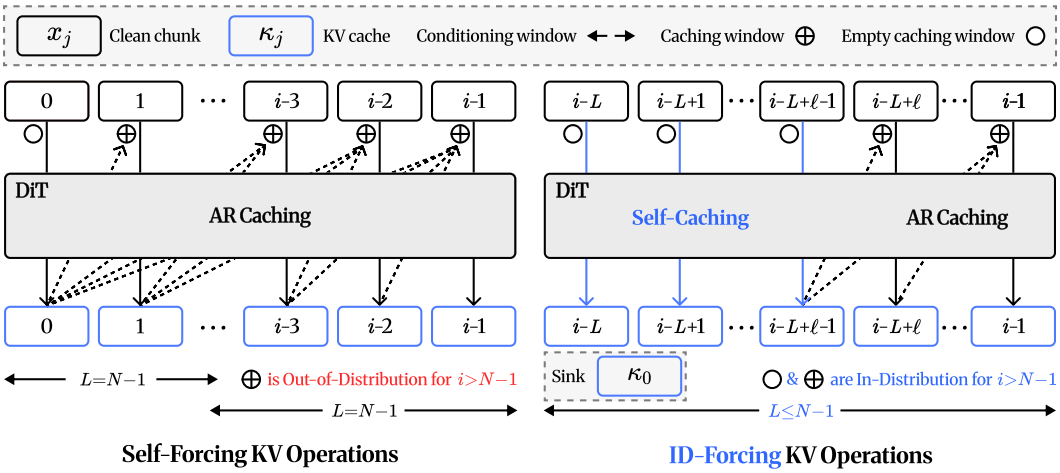


Figure 3: **KV operations comparison.** Both illustrations show the caching window of each KV entry and the conditioning window of size L used to generate chunk x_i. **Left**: Self-Forcing caches every entry autoregressively, so entries beyond the horizon are cached under windows unseen in training. **Right**: ID-Forcing caches the earlier entries with self-caching and the rest autoregressively on top of them, so every entry is cached under a window seen in training.

KV conditioning (Eq. (2)) and KV caching (Eq. (3)) are therefore unchanged in form, but both now attend to the L most recent KV entries $\kappa_{i-L:i-1}$ rather than the full prefix $\kappa_{<i}$. This manifests as quality degradation known as *drifting* (exposure bias) (Chen et al., 2024; Huang et al., 2025b): colors and textures shift, and motion dynamics decay.

Temporal Positions. Attention uses rotary position embeddings (RoPE (Su et al., 2024)); we consider only the temporal axis in this work. Rotations are applied to queries and keys, and attention logits depend only on the relative temporal gap between them, not on their absolute indices. A chunk occupies a *slot* in the attention window, and each slot receives the f consecutive temporal indices. We write R_Δ for re-rotation of a KV entry's stored keys by Δ frames.

3 In-Distribution Forcing

We propose **In-Distribution Forcing (ID-Forcing)**, a training-free framework that keeps test-time extrapolation within the training distribution. Prior works address drifting by modifying KV conditioning, treating drifting as a consequence of discarding informative past context (Yi et al., 2026; Yesiltepe et al., 2026; Kim et al., 2026). We instead view drifting as a deviation from the training distribution, and show that this deviation occurs not just in KV conditioning, but also in KV caching. ID-Forcing therefore introduces two-level control over KV operations: KV caching and KV conditioning.

Level 1 (KV caching): How each KV entry is written (§3.1).
Level 2 (KV conditioning): How KV entries are read during chunk generation (§3.2).

3.1 Level 1 – KV Caching: Enforcing In-Distribution Provenance

Our key observation is that the same clean chunk x_i maps to different points in KV space depending on its *provenance*, the existing KV entries within the caching window (Fig. 2). Crucially, when this provenance involves unseen context, the resulting KV entry κ_i falls out-of-distribution (OOD), and we term this the *KV-provenance problem*.

Beyond the horizon, a rolling window introduces the KV-provenance problem. Subsequently, every later generation that reads the resulting KV entries inherits their OOD state, which accumulates as drifting. To resolve this problem, we propose **Self-Caching**, which keeps KV caching strictly within the training distribution throughout the entire extrapolation.

Motivation. We first inspect the generation process (Eq. (2)) of an arbitrary beyond-horizon chunk x_i. At its first denoising step, x_i is initialized from pure noise, *i.e.*, $x_i^T \sim \mathcal{N}(0, I)$, and its queries, keys, and values are generated via frozen projections. Consequently, the input itself remains in-distribution, leaving the KV conditioning window as the only possible entry point for drift.

> **Observation 1.** *For every $i \geq N$, the KV entries in the conditioning window are the sole path through which drift can enter the generation of chunk x_i.*

The question is then whether those KV entries are themselves in-distribution. This depends on the caching window each of them attended to when cached, *i.e.*, its provenance.

> **Observation 2.** *The KV entries read when generating chunk x_i are in-distribution only if each of them was cached under a provenance that training executed.*

The two observations show that generating x_i in-distribution requires in-distribution KV entries in the conditioning window. These entries, in turn, are in-distribution only if their provenance is.

Self-Caching. We propose a simple caching rule that keeps the provenance of each KV entry in-distribution throughout the extrapolation. Concretely, we restrict every provenance to the configurations that training executed, so that no KV entry is ever written under an unseen context.

For a conditioning window of L KV entries, we cache the first ℓ entries attending only to themselves, $E(x_j; \varnothing)$. The remaining $L-\ell$ entries are then cached autoregressively on top of the last self-cached entry, mirroring how training caches $\kappa_1, \kappa_2, \ldots$ on top of κ_0. Fig. 3 visualizes the proposed method.

KV caching rule. For every $i \geq N$, the conditioning window of length $L < N$ consists of the KV entries $\kappa_{i-L}, \ldots, \kappa_{i-1}$, which are cached as:

$$\kappa_j = \begin{cases} E\big(x_j; \varnothing\big), & i{-}L \leq j < i{-}L{+}\ell, \\ E\big(x_j; \kappa_{i-L+\ell-1\,:\,j-1}\big), & i{-}L{+}\ell \leq j \leq i{-}1. \end{cases} \tag{4}$$

Starting from the training horizon, every chunk is generated from in-distribution KV entries and is itself cached as an in-distribution one, so merely depending on earlier chunks does not cause drift.

3.2 Level 2 – KV Conditioning: Enabling Exact Rolling Window

Self-caching keeps every stored KV entry in-distribution, but how these entries are read must also match training. We address KV conditioning from two perspectives: (1) the distributional role of the first chunk, and (2) the exact rolling window enabled by self-caching.

First Chunk as a Persistent Condition. During training, every generation is conditioned on κ_0, meaning the conditioning window always contains the first chunk within the training horizon. However, as the window rolls during extrapolation, κ_0 is evicted; yet κ_0 is in fact itself a necessary condition for staying in-distribution at extrapolation.

Unlike later chunks, the causal 3D-VAE (Wan et al., 2025) encodes the initial latent frame from a single pixel frame, whereas every subsequent frame aggregates multiple frames. κ_0 is therefore distributionally irreplaceable, and evicting it once the window rolls ($i \geq N$) is itself out-of-distribution. This differs from an attention sink (Xiao et al., 2024; Yi et al., 2026) in both motivation and form. Attention sinks retain several early KV entries for their empirical effect of stabilizing generation, whereas we retain exactly the first one because it alone is distributionally unique.

Algorithm 1 In-Distribution Forcing

```
# G: T-step denoiser, E: KV caching
# N, M: training, target horizon
# L: window size, n_self: self-cached

X, KV = horizon_chunks(), self_cached_KV()

for i in range(N, M):
   # sink + L-1 recent, re-rotated
   W = [KV[0]] + rerope(KV[i-L+1:i])

   # re-cache the tail autoregressively
   for j in range(n_self, L):
      W[j] = E(X[i-L+j], W[n_self-1:j])

   # KV conditioning
   x = randn(*shape)
   for t in range(T, 0, -1):
      x = G(x, t, W)

   # Self-caching
   X.append(x)
   KV.append(E(x, []))

return X
```

Bounded Window and Exact Rolling. We constrain our conditioning window to a length of $L < N$, holding κ_0 at slot 0 and the $L-1$ most recent KV entries in the remaining slots. This strictly bounds the total window size within the training budget $N-1$. After generating each chunk, the oldest non-sink KV entry is evicted and the remaining ones are re-rotated by R_{-f}, while κ_0 remains fixed at slot 0. Consequently, every temporal distance within the window matches a state observed during training.

Self-caching is what keeps this rolling window exactly in-distribution. During training, every cached KV entry coexists with its complete provenance set because the training window only appends entries without eviction. Under test-time rolling, however, eviction removes past context, leaving the remaining KV entries without their original provenance. By contrast, a self-cached KV entry possesses no external provenance to lose; thus, eviction leaves its representation strictly within the training distribution. The complete procedure of our method is summarized in Algorithm 1.

KV conditioning rule. For every $i \geq N$, the oldest KV entry other than κ_0 is evicted and the remaining KV entries are re-rotated by R_{-f} before chunk i reads them:

$$\{\kappa_0\} \cup \{\kappa_{i-L+1}, \ldots, \kappa_{i-1}\} \;\leftarrow\; \{\kappa_0\} \cup R_{-f}\big[\{\kappa_{i-L+1}, \ldots, \kappa_{i-1}\}\big]. \tag{5}$$

4 Experiments

We evaluate ID-Forcing on minute-scale video generation, extrapolating far beyond the training horizon on two base models, Self-Forcing (Huang et al., 2025b) and LongLive (Yang et al., 2026). In §4.2, we compare ID-Forcing with existing test-time KV conditioning methods on 120- and 240-second generation through quantitative metrics, qualitative comparisons, and a user study. In §4.3, we examine how each component of ID-Forcing; self-caching for KV caching, and the first-chunk sink and re-rotation for KV conditioning, contributes to suppressing drifting.

4.1 Experimental Setup

Implementation Details. We implement our method on top of the Wan2.1-T2V-1.3B (Wan et al., 2025) architecture and evaluate it across two base models: Self-Forcing (Huang et al., 2025b) and LongLive (Yang et al., 2026). We compare against prior test-time extrapolation methods for autoregressive video diffusion, Deep Forcing (Yi et al., 2026), ∞-RoPE (Yesiltepe et al., 2026), and MemRoPE (Kim et al., 2026). Each video chunk consists of 3 latent frames, generated using a 4-step denoising schedule at timesteps $\{1000, 750, 500, 250\}$. We use conditioning window length $L = 3$ and self-caching length $\ell = 2$ for Self-Forcing. For LongLive, which is trained on minute-long videos, we apply our method from the first chunk with $L = 3$ and $\ell = 3$. In both cases, ℓ counts the sink κ_0.

Evaluation. We generate 120- and 240-second videos at a resolution of 480×832 using the prompts from MovieGen (Polyak et al., 2024). Following previous works (Yin et al., 2025; Cui et al., 2026; Yi et al., 2026; Kim et al., 2026), we use the first 128 prompts which are refined using Qwen2.5-7B-Instruct. We report metrics from the VBench-Long (Huang et al., 2024) benchmark, along with drift-specific metrics: color drift (Xiang et al., 2026) and motion drift. Details for these metrics are provided in App. §B. To further validate automated metrics, we conduct a user study following the 2AFC protocol (Yi et al., 2026; Kim et al., 2026).

4.2 Main Results

Quantitative Results. Tab. 1 reports VBench-Long (Huang et al., 2024) metrics for 120- and 240-second videos, built on top of two base models. Compared with all baselines, ID-Forcing achieves substantial gains on both drift metrics, indicating that visual quality and motion dynamics remain stable over long durations. This resistance to motion degradation is further shown by Dynamic Degree, where our method achieves the highest score in every setting. In contrast, base models such as Self-Forcing and LongLive exhibit markedly lower dynamic degree, particularly at longer durations, reflecting a tendency toward frame freezing as generation extends.

Notably, our method also performs well on Aesthetic Quality, Imaging Quality, and Subject Consistency. The one exception is Motion Smoothness, which measures local frame-to-frame stability

Table 1: **Quantitative evaluation on long video generation.** We report VBench-Long (Huang et al., 2024) metrics and drift metrics (color drift and motion drift) on 120- and 240-second videos, with methods grouped by base model.

Method	Aesthetic Quality ↑	Background Consistency ↑	Imaging Quality ↑	Motion Smoothness ↑	Subject Consistency ↑	Dynamic Degree ↑	Color Drift ↑	Motion Drift ↑
120 seconds								
Results on Self-Forcing								
Self-Forcing	49.95	96.17	61.60	**98.26**	96.20	30.04	25.54	46.88
Deep Forcing	57.26	**96.28**	65.94	98.21	**97.22**	45.49	59.65	71.88
∞-RoPE	55.54	95.58	67.42	97.64	96.18	62.30	54.15	85.16
MemRoPE	55.45	95.78	67.70	97.69	96.32	62.90	59.25	82.81
ID-Forcing (Ours)	**58.23**	96.13	**68.20**	97.77	96.76	**65.65**	**72.37**	**89.06**
Results on LongLive								
LongLive	**59.25**	**96.52**	67.32	**98.68**	97.57	42.55	59.98	91.41
Deep Forcing	58.31	96.48	66.12	98.49	97.55	45.14	60.24	89.84
∞-RoPE	56.90	96.22	66.53	98.53	97.12	52.81	58.20	95.31
MemRoPE	57.63	96.39	68.40	98.65	97.30	49.18	65.49	**96.88**
ID-Forcing (Ours)	58.99	96.48	**68.73**	98.57	**97.87**	**55.01**	**70.94**	96.09
240 seconds								
Results on Self-Forcing								
Self-Forcing	44.90	**96.21**	58.38	**97.97**	95.59	29.61	20.98	56.25
Deep Forcing	56.27	95.93	65.63	97.83	96.59	58.82	62.52	88.28
∞-RoPE	55.39	95.61	67.48	97.61	96.17	61.98	55.49	87.50
MemRoPE	54.29	95.83	67.36	97.96	96.47	59.75	61.41	85.16
ID-Forcing (Ours)	**57.72**	96.10	**67.88**	97.62	**96.70**	**64.76**	**70.34**	**92.19**
Results on LongLive								
LongLive	58.94	**96.58**	67.88	**98.74**	**97.64**	39.79	55.64	85.16
Deep Forcing	57.91	96.27	66.61	98.40	97.17	46.02	57.08	89.06
∞-RoPE	56.63	96.14	67.07	98.48	96.97	53.95	56.72	94.53
MemRoPE	57.64	96.35	68.58	98.66	97.30	48.76	64.23	**96.88**
ID-Forcing (Ours)	**60.44**	96.40	**68.79**	98.50	97.57	**56.13**	**71.76**	96.09

and is therefore inherently biased toward methods with lower dynamic degree; on all other metrics, ID-Forcing remains competitive with or superior to the baselines. This result suggests that our method mitigates long-horizon drift without sacrificing generation quality.

Qualitative Results. Fig. 4 shows qualitative comparisons on 2-minute video generation. On the Self-Forcing base model, the base model itself suffers severe color and motion collapse over time. Adding Deep Forcing stabilizes color but subject consistency still degrades, with the puppies' identities gradually drifting. ∞-RoPE exhibits color saturation alongside a similar loss of subject consistency, while MemRoPE shows unnatural color shifts and changing textures across frames. In contrast, ID-Forcing remains stable throughout, preserving both subject consistency and color fidelity.

On the LongLive base model, color drift is generally less severe across all methods, but subject consistency remains a challenge: the base model shows the subject's clothing color changing over time. Deep Forcing suppresses motion and introduces color artifacts, including unnatural brightening and a bluish tint. Both ∞-RoPE and MemRoPE exhibit highly inconsistent motion throughout generation. ID-Forcing, by contrast, maintains subject consistency, sustains motion until the end of the sequence, and preserves accurate color throughout.

User Study. We conducted a user study to complement our quantitative results. Following the 2AFC protocol used in prior works (Yi et al., 2026; Kim et al., 2026), we extend both the evaluation metrics and the number of prompts to specifically capture long-horizon degradation. 20 participants compared our method against 4 Self-Forcing-based baselines across 20 diverse prompts, each judging a single Baseline-vs-Ours pair per prompt across six dimensions: Color Consistency, Background Consistency, Subject Consistency, Dynamic Degree, Temporal Flickering, and Overall Preference (details in App. §C). As shown in Tab. 2, participants consistently preferred our method over all baselines across every evaluated dimension.

Table 2: **User study.** Preference rate (%) of ID-Forcing over each baseline in pairwise comparisons (Baseline vs. Ours). Each value denotes the percentage of participants who preferred ours (↑).

Method	Color Cons.	Bg. Cons.	Subj. Cons.	Dyn. Deg.	Temp. Flick.	Over. Pref.
Self-Forcing	100.0	99.0	96.0	93.0	98.0	100.0
Deep Forcing	90.0	84.0	85.0	81.0	97.0	95.0
∞-RoPE	91.0	90.0	87.0	75.0	91.0	93.0
MemRoPE	88.0	82.0	78.0	59.0	89.0	86.0



Figure 4: **Qualitative comparison on 2-minute videos.** Full videos are available at this link.

4.3 Ablation Studies

We ablate the three components of ID-Forcing: self-caching for KV caching, and, for KV conditioning, retaining the first chunk as a sink and re-rotating the window into the trained positional range. As shown in Fig. 5, without any of them, Self-Forcing drastically collapses beyond the training horizon. Retaining the first chunk largely stabilizes generation, consistent with the known empirical effect of attention sinks, but colors and textures still drift, frames flicker, and the video periodically resets to an earlier scene. Adding re-rotation improves this further, yet drifting persists, as KV caching remains out-of-distribution. Self-caching brings KV caching in-distribution as well, greatly reducing color and texture drifting while preserving the dynamics. Quantitative results are in Tab. 4 of App. §A.

5 Related Works

Video Diffusion Models. Video diffusion models with bidirectional attention produce high-quality clips (OpenAI, 2025; Wan et al., 2025; Kong et al., 2024; Yang et al., 2025), but attend over the

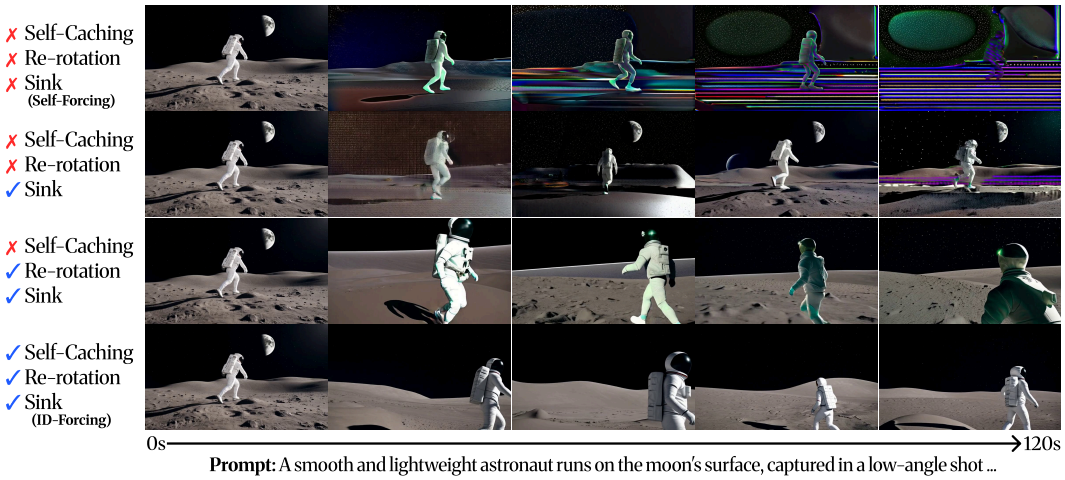


Figure 5: **Qualitative ablation on 2-minute video.** Components are added from top to bottom. Without self-caching, colors and textures still drift even with the sink and re-rotation, whereas the full ID-Forcing preserves the scene and color throughout the 2-minute video.

entire sequence at once, so their cost grows quadratically with length and the output length is fixed at sampling time. Autoregressive (AR) formulations instead factorize the video into chunks generated in sequence, each conditioned on the KV cache of its predecessors (Chen et al., 2024; Yin et al., 2025; Huang et al., 2025b). This makes the cost linear, admits streaming output, and in principle allows a video of arbitrary length to be produced at test time. The AR factorization, however, feeds each chunk back as the condition for the next, so any deviation from the training distribution accumulates along the rollout, a failure known as *drifting* (Chen et al., 2024; Huang et al., 2025b).

Addressing Drifting in AR Video Diffusion. The most straightforward way is to train on longer contexts (Cui et al., 2026; Liu et al., 2026; Yang et al., 2026; Chen et al., 2026; Lu et al., 2026; Chen et al., 2025; Teng et al., 2025). However, this does not remove the boundary, since training always covers a fixed horizon while the AR formulation exists to continue past whatever length was trained. Moreover, this is infeasible with extremely long videos since the cost of extending that horizon during training grows with the target length. A test-time extrapolation method is therefore necessary regardless of how the model was trained, and the two directions are orthogonal: a better-trained backbone still requires a rule for what to do beyond its horizon.

Test-time methods approach this by managing the KV cache during generation. Deep Forcing (Yi et al., 2026) keeps early KV entries as sinks, realigns their temporal RoPE to the current timeline, and compresses the rest. ∞-RoPE (Yesiltepe et al., 2026) reassigns temporal indices so that the offsets seen at inference stay within the trained range. MemRoPE (Kim et al., 2026) caches keys without rotation and aggregates evicted entries into memory tokens, applying RoPE online at attention time. All of these act on *KV conditioning*, leaving untouched how each KV entry was written. In contrast, we address both *KV caching* and *KV conditioning*, keeping test-time extrapolation within the training distribution.

6 Conclusion

In this paper, we introduce **In-Distribution Forcing**, a training-free framework designed to enable stable long-horizon video generation. By analyzing the autoregressive (AR) model's key-value (KV) dynamics, we identify the *KV-provenance problem*: a critical out-of-distribution (OOD) failure mode originating at the caching level. To resolve this, we propose a two-level KV management rule: at the caching level, we apply self-caching for KV caching; at the conditioning level, we strictly bound the window length ($L \leq N-1$) while permanently fixing the first chunk κ_0. Extensive quantitative and qualitative evaluations demonstrate that our method achieves highly competitive performance across diverse durations, backbones, and metrics. Most importantly, our framework effectively addresses the quality drift issue inherent to autoregressive models, maintaining stable video dynamics even over minute-scale generations.

AI USE STATEMENT

We did not use generative AI tools for any tasks with required disclosure under the ICLR 2027 policy. We used generative AI tools only to improve the readability of the manuscript, such as polishing sentences and checking grammar. All AI-assisted edits were reviewed by the authors, and we take full responsibility for the content of this work.

REFERENCES

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *NeurIPS*, 2024.

Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, et al. Skyreels-v2: Infinite-length film generative model. *arXiv:2504.13074*, 2025.

Shuo Chen, Cong Wei, Sun Sun, Tiancheng SHEN, Ping Nie, Kai Zhou, Ge Zhang, Ming-Hsuan Yang, and Wenhu Chen. Context forcing: Consistent autoregressive video generation with long context. In *ICML*, 2026.

Justin 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 *ICLR*, 2026.

Mohamed Elmoghany, Liangbing Zhao, Xiaoqian Shen, Subhojyoti Mukherjee, Yang Zhou, Gang Wu, Viet Dac Lai, Seunghyun Yoon, Ryan Rossi, Abdullah Rashwan, Puneet Mathur, Varun Manjunatha, Daksh Dangi, Chien Nguyen, Nedim Lipka, Trung Bui, Krishna Kumar Singh, Ruiyi Zhang, Xiaolei Huang, Jaemin Cho, Yu Wang, Namyong Park, Zhengzhong Tu, Hongjie Chen, Hoda Eldardiry, Nesreen Ahmed, Thien Nguyen, Dinesh Manocha, Mohamed Elhoseiny, and Franck Dernoncourt. Infinitystory: Unlimited video generation with world consistency and character-aware shot transitions. *arXiv:2603.03646*, 2026.

Google DeepMind. Veo: A text-to-video generation system. Technical report, Google DeepMind, 2025. https://storage.googleapis.com/deepmind-media/veo/Veo-3-Tech-Report.pdf.

Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Streamingt2v: Consistent, dynamic, and extendable long video generation from text. In *CVPR*, 2025.

Yicong Hong, Yiqun Mei, Chongjian Ge, Yiran Xu, Yang Zhou, Sai Bi, Yannick Hold-Geoffroy, Mike Roberts, Matthew Fisher, Eli Shechtman, Kalyan Sunkavalli, Feng Liu, Zhengqi Li, and Hao Tan. Relic: Interactive video world model with long-horizon memory. *arXiv:2512.04040*, 2025.

Tianyu Huang, Wangguandong Zheng, Tengfei Wang, Yuhao Liu, Zhenwei Wang, Junta Wu, Jie Jiang, Hui Li, Rynson W.H. Lau, Wangmeng Zuo, and Chunchao Guo. Voyager: Long-range and world-consistent video diffusion for explorable 3d scene generation. *arXiv:2506.04225*, 2025a.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *NeurIPS*, 2025b.

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 *CVPR*, 2024.

Youngrae Kim, Qixin Hu, C.-C. Jay Kuo, and Peter A. Beerel. MemRoPE: Training-free infinite video generation via evolving memory tokens. In *ECCV*, 2026.

Kling Team, Kuaishou Technology. Kling-omni technical report. *arXiv:2512.16776*, 2025.

Akio Kodaira, Tingbo Hou, Ji Hou, Markos Georgopoulos, Felix Juefei-Xu, Masayoshi Tomizuka, and Yue Zhao. Streamdit: Real-time streaming text-to-video generation. In *CVPR*, 2026.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv:2412.03603*, 2024.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. In *ICLR*, 2026.

Yunhong Lu, Yanhong Zeng, Haobo Li, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jiapeng Zhu, Hengyuan Cao, Zhipeng Zhang, Xing Zhu, et al. Reward forcing: Efficient streaming video generation with rewarded distribution matching distillation. In *CVPR*, 2026.

Yawen Luo, Xiaoyu Shi, Junhao Zhuang, Yutian Chen, Quande Liu, Xintao Wang, Pengfei Wan, and Tianfan Xue. Shotstream: Streaming multi-shot video generation for interactive storytelling. In *ECCV*, 2026.

OpenAI. Sora 2 system card. Technical report, OpenAI, 2025. https://openai.com/index/sora-2-system-card/.

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

Tianchang Shen, Sherwin Bahmani, Kai He, Sangeetha Grama Srinivasan, Tianshi Cao, Jiawei Ren, Ruilong Li, Zian Wang, Nicholas Sharp, Zan Gojcic, Sanja Fidler, Jiahui Huang, Huan Ling, Jun Gao, and Xuanchi Ren. Lyra 2.0: Explorable generative 3d worlds. In *SIGGRAPH Asia*, 2026.

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

Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv:2505.13211*, 2025.

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

Xunzhi Xiang et al. Pathwise test-time correction for autoregressive long video generation. *arXiv:2602.05871*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 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 *ICLR*, 2026.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *ICLR*, 2025.

Hidir Yesiltepe, Tuna Han Salih Meral, Adil Kaan Akan, Kaan Oktay, and Pinar Yanardag. Infinity-RoPE: Action-controllable infinite video generation emerges from autoregressive self-rollout. In *CVPR*, 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 *ICML*, 2026.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Frédo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *CVPR*, 2025.

APPENDIX

Table 3: Quantitative evaluation on VBench-Long (Huang et al., 2024) (30s and 60s).

Method	Aesthetic Quality ↑	Background Consistency ↑	Imaging Quality ↑	Motion Smoothness ↑	Subject Consistency ↑	Dynamic Degree ↑	Color Drift ↑	Motion Drift ↑
30 seconds								
Results on Self-Forcing								
Self-Forcing	56.85	96.06	67.48	**98.37**	96.33	44.58	37.45	60.94
Deep Forcing	57.79	96.05	67.00	98.13	**96.79**	59.22	63.61	82.03
∞-RoPE	57.38	95.87	67.84	97.80	96.65	64.79	56.77	85.94
MemRoPE	57.32	96.08	67.73	98.05	96.78	**66.35**	65.61	89.84
ID-Forcing (Ours)	**58.40**	**96.15**	**68.20**	97.67	96.70	65.94	**72.13**	**95.31**
Results on LongLive								
LongLive	59.03	96.64	68.15	98.72	97.75	45.10	59.65	94.53
Deep Forcing	58.91	**96.80**	68.20	98.72	**98.06**	43.75	65.81	91.41
∞-RoPE	58.41	96.54	67.78	98.66	97.57	48.54	59.21	**96.88**
MemRoPE	58.83	96.55	68.74	98.69	97.56	46.98	63.61	91.41
ID-Forcing (Ours)	**60.61**	96.60	**69.01**	98.67	97.89	**49.48**	**73.14**	95.31
60 seconds								
Results on Self-Forcing								
Self-Forcing	54.91	96.06	65.60	97.93	96.33	36.82	30.61	53.12
Deep Forcing	57.65	**96.33**	67.03	**98.31**	**97.21**	50.47	65.07	74.22
∞-RoPE	56.16	95.69	67.76	97.67	96.46	63.25	55.87	83.59
MemRoPE	56.45	95.98	66.64	97.88	96.68	63.46	63.43	89.84
ID-Forcing (Ours)	**58.69**	96.15	**68.41**	97.90	96.80	**65.83**	**71.91**	**92.97**
Results on LongLive								
LongLive	**59.47**	96.60	68.09	**98.73**	97.60	44.11	59.69	93.75
Deep Forcing	58.96	**96.76**	67.27	98.70	97.85	45.39	62.89	92.19
∞-RoPE	57.97	96.47	67.38	98.65	97.38	50.44	59.77	**96.88**
MemRoPE	58.25	96.48	68.22	98.71	97.46	47.32	60.97	92.97
ID-Forcing (Ours)	59.33	96.51	**68.95**	98.61	**97.87**	**53.88**	**71.76**	96.09

A ADDITIONAL RESULTS

Quantitative Results. We provide additional VBench-Long (Huang et al., 2024) metrics on 30- and 60-second videos to complement the 120- and 240-second results reported in the main paper. Consistent with our main findings, ID-Forcing achieves substantial improvements on both Color Drift and Motion Drift, and likewise attains the highest Dynamic Degree among all compared methods.

Comparing across all four durations (30, 60, 120, and 240 seconds), several baselines exhibit a clear decline in performance as generation length increases, particularly on Dynamic Degree and the drift metrics (*e.g.* Self-Forcing Dynamic Degree collapses from **44.58** at 30 seconds to **29.61** at 240 seconds), reflecting a growing tendency toward frame freezing and quality degradation over longer horizons. In contrast, ID-Forcing maintains largely consistent performance across all durations (*e.g.* **65.94** at 30 seconds to **64.76** at 240 seconds), indicating that our method is robust to extrapolation and does not suffer from the same long-horizon degradation observed in the baselines.

Qualitative Results. We provide additional qualitative comparisons on 2-minute video generation in Fig. 6. Additional video results spanning all 4 durations (30s, 60s, 120s, 240s), including our paper reported results, can be found here: `https://idforcing.github.io`.

Ablation Results. Tab. 4 reports the quantitative evaluations of ablation study, measured on Self-Forcing at 120 seconds with components added cumulatively.

Self-Forcing scores highest on Background Consistency and Motion Smoothness, but this reflects its collapse rather than its quality: once the video degenerates, consecutive frames become nearly identical, which both metrics reward. Its Dynamic Degree of 30.04 and Color Drift of 25.54 show what the frames make visible.

Adding the sink raises Dynamic Degree and Color Drift substantially, confirming that generation is stabilized, yet Color Drift remains at 48.65 and Motion Drift at 67.19, consistent with the drifting and periodic resets still visible in Fig. 5. Re-rotation improves Dynamic Degree and both drift metrics further, but Color Drift stays at 60.66, since KV caching is still out of distribution.

Table 4: Quantitative results of ablation study on VBench-Long (Huang et al., 2024) (120s).

Method	Aesthetic Quality ↑	Background Consistency ↑	Imaging Quality ↑	Motion Smoothness ↑	Subject Consistency ↑	Dynamic Degree ↑	Color Drift ↑	Motion Drift ↑
Self-Forcing	49.95	**96.17**	61.60	**98.26**	96.20	30.04	25.54	46.88
+ Sink	55.23	95.59	64.68	96.88	95.70	46.90	48.65	67.19
+ Re-rotation	54.14	95.68	66.89	97.36	96.08	65.44	60.66	87.50
+ Self-Caching **(ID-Forcing)**	**58.23**	96.13	**68.20**	97.77	**96.76**	**65.65**	**72.37**	**89.06**

With self-caching, ID-Forcing yields the best score on every metric except Background Consistency and Motion Smoothness, which favor the collapsed output of Self-Forcing. Color Drift improves from 60.66 to 72.37 and Aesthetic Quality from 54.14 to 58.23, matching the reduced drifting and preserved dynamics observed in the frames.

B Metric Description

Color Drift (HSV Histogram). Following Xiang et al. (2026), we quantify changes in color distribution over the course of a generated sequence by comparing the color histograms of the first and last frames. Given the initial frame and final frame of a generated video, we convert both to HSV space and compute an L_1-normalized histogram of the Hue channel using 180 bins, denoted $h_{\text{start}}, h_{\text{end}} \in \mathbb{R}^{180}$, with $\|h_{\text{start}}\|_1 = \|h_{\text{end}}\|_1 = 1$. We measure the L_1 distance $\|h_{\text{start}} - h_{\text{end}}\|_1$ between the two histograms, which lies in $[0, 2]$.

Motion Drift (Dynamic Degree Difference). To measure how well motion is sustained throughout a generated sequence, we compute the Dynamic Degree (Huang et al., 2024) separately over the first 5 seconds and the last 5 seconds of the video, denoted d_{start} and d_{end}, respectively. We measure the difference $|d_{\text{start}} - d_{\text{end}}|$, which lies in $[0, 1]$ and captures the extent to which motion established early in the video is preserved or lost by the end of generation. A large difference reflects degradation, most commonly a collapse toward static, near-frozen frames.

Score Calibration. For consistency with VBench-Long metrics, where higher is better and scores range from 0 to 100, we convert both drift measures into scores by normalizing each distance by its maximum and subtracting it from 1:

$$\text{Color Drift} = 100\left(1 - \tfrac{1}{2}\|h_{\text{start}} - h_{\text{end}}\|_1\right), \qquad \text{Motion Drift} = 100\left(1 - |d_{\text{start}} - d_{\text{end}}|\right). \tag{6}$$

Both scores are computed for each generated video and averaged over all 128 prompts. A higher score thus indicates that color and motion dynamics remain more stable throughout generation. All drift results in the main paper are reported in this calibrated form.

C User Study Details

For each of the 20 prompts, participants were shown two anonymized videos side by side: one generated by our method and one by a randomly assigned baseline. They were asked to indicate their preference along Color Consistency, Background Consistency, Subject Consistency, Dynamic Degree, Temporal Flickering, and Overall Preference. Fig. 7 demonstrates our user study interface and Fig. 8 show our criteria explanation page.

To ensure fair comparisons across all baselines, we employed a balanced Latin-square design for baseline assignment. We divided the 20 prompts into 5 groups of 4 and split participants into 4 groups. Within each group of 4 prompts, we rotated the baseline assignment so that each of the 4 baselines appeared exactly once. As a result, every participant saw each baseline exactly 5 times (5 groups × 1 time each), and every prompt was compared against all 4 baselines equally across the full set of participants. This ensured that no baseline was unfairly matched with easier or harder prompts, keeping all comparisons balanced.



Figure 6: Additional Qualitative results on 2-minute videos.

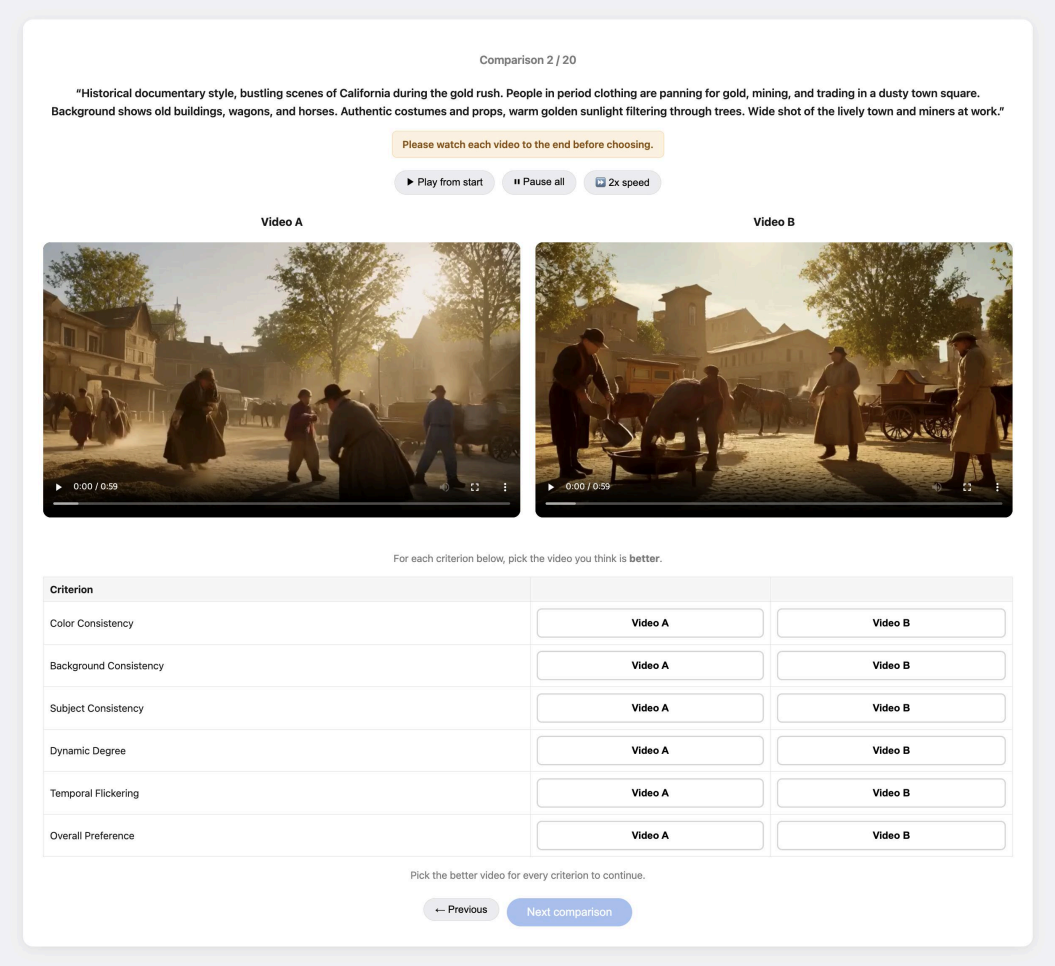


Figure 7: User study interface.

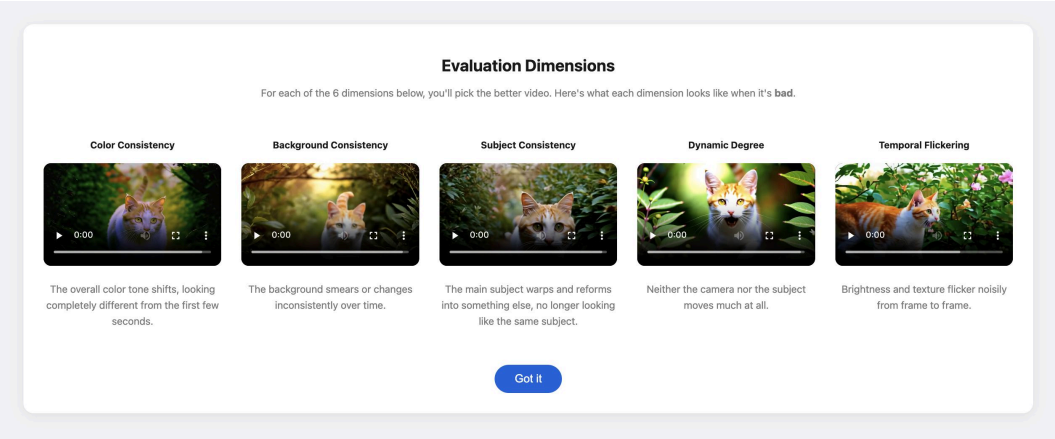


Figure 8: Criteria explanation.