AdaState: Self-Evolving Anchors for Streaming Video Generation

Yusuf Dalva **Pinar Yanardag**
{ydalva, pinary}@vt.edu
Virginia Tech
adastate.github.io

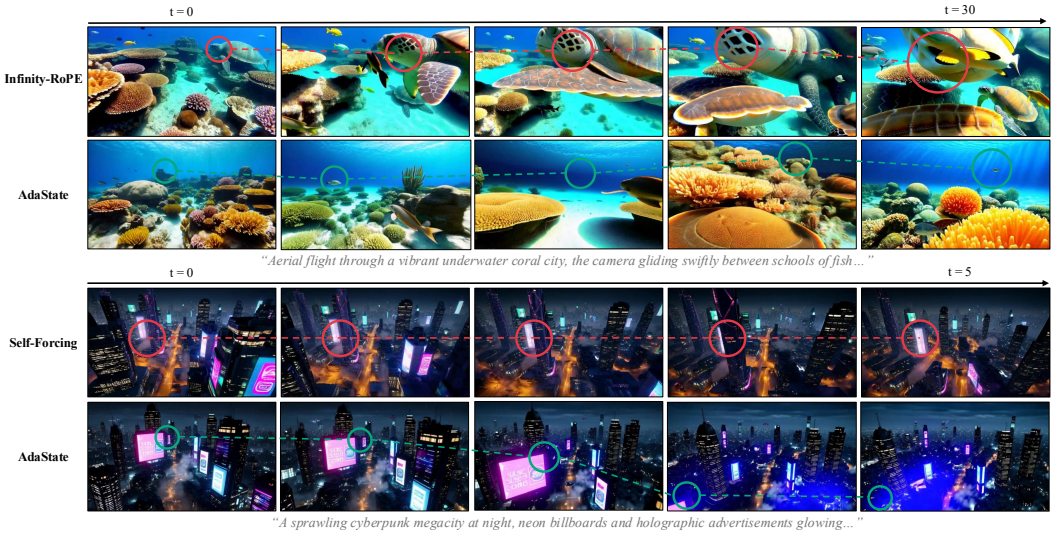


Figure 1: **AdaState.** Colored markers and dashed lines track scene progression (red: baselines; teal: AdaState). *Top, 30s (6× training horizon):* Infinity-RoPE's static anchor prevents the scene from evolving, forcing fish and turtles into the initial layout and producing duplications. AdaState allows the scene to evolve, revealing new terrain and subjects throughout the rollout. *Bottom, 5s:* Self-Forcing holds the scene in place as neon lighting dims and the cityscape desaturates; AdaState produces continuous scene evolution while preserving visual characteristics.

Abstract

Autoregressive video diffusion models generate streaming video by producing frames sequentially, conditioning each chunk on previously generated content. These models are structurally anchored to the first frame: its key-value representation occupies a privileged position in the attention cache and serves as the primary scene reference throughout generation. As the cleanest and most error-free position in the cache, this anchor draws disproportionate attention, suppressing video dynamics, and locking scene composition to the initial viewpoint even as the scene naturally evolves. The result is a temporally shallow video in which motion, camera movement, and scene progression are dampened in favor of static consistency. To address this, we replace the static anchor with an *adaptive state*, a hidden latent that the model denoises alongside content at every chunk but never renders. Rather than referencing a frozen first frame, the model generates its own scene anchor at each step by attending to both the previous state and the current content, producing

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

a reference that evolves with the generated content. Unlike standard video generation, which encodes an absolute notion of time, our formulation treats time as relative: every generation step sees the same positional structure regardless of how far generation has progressed, and the state transition is identical at every chunk. Together, these properties introduce a recurrence into the generation process, where denoising serves as the transition function, and the KV cache serves as the carrier, requiring no external module. Experiments demonstrate that the adaptive state substantially improves video dynamics, enabling richer motion and natural scene progression within generated videos.

1 Introduction

Autoregressive video diffusion models generate streaming video by producing one chunk of frames at a time, conditioning each chunk on previously generated content [12, 35, 31, 17, 33]. These models are structurally anchored to the first frame: its key-value representation occupies a dominant position in the attention cache and serves as the primary scene reference throughout generation, exploiting the attention-sink phenomenon [29] where causal softmax concentrates mass on initial positions. However, as the cleanest and most error-free position in the cache, this static anchor draws disproportionate attention, suppressing video dynamics and locking scene composition to the initial viewpoint even as the scene naturally evolves during generation. Motion, camera movement, and scene progression are dampened in favor of static consistency, producing temporally shallow video that lacks the dynamic richness of natural video (see Figure 1). The static anchor also limits the model's trained robustness, since errors accumulated over the autoregressive rollout are absorbed by the clean reference rather than surfaced at the model's most attended position, leaving the training objective with limited leverage to shape behavior under the imperfect conditions the model will encounter as generation extends.

Existing approaches do not address this root cause: static sinks [31, 17] retain first-frame tokens as fixed anchors, reinforcing the shortcut. EMA-based methods [18, 14] apply content-agnostic averaging over evicted content, converging to a blurry mean that cannot adapt to scene changes. Token replacement approaches [16] substitute raw cached content into the anchor position on a heuristic schedule, keeping the reference fresh but biasing generation toward reproducing past frames rather than producing novel continuation. All of these either preserve the static anchor or update it through operations external to the generative model, leaving the fundamental attention bias intact.

In this paper, we replace the static anchor with an *adaptive state*, a hidden latent that the model denoises alongside content at every chunk but never renders. At each step, the model generates its own scene anchor by attending to both the current content and the previous adaptive state, producing a reference that evolves with the generated content rather than remaining locked to the initial frame. After denoising, the state's clean representation is written to the anchor position in the cache, while the content is decoded into video; the state is carried forward silently as an evolving scene reference. Unlike standard video generation, which encodes absolute temporal position, our formulation treats time as relative: every generation step sees the same positional structure regardless of how far generation has progressed, removing the notion of a privileged time zero. This design reveals that denoising is itself a recurrence: the adaptive state is a hidden variable updated by the model's own iterative refinement and carried via the KV cache, turning sequential autoregressive generation into a recurrent process with no external module or gating mechanism. To ensure the training objective emphasizes the frames that depend most on the adaptive state, we further propose horizon-weighted DMD, a per-frame loss weighting that increases with frame index, preventing the optimizer from concentrating capacity on clean, early frames. Experiments demonstrate that the adaptive state substantially improves video dynamics, enabling richer motion and natural scene progression within generated videos. **Our contributions:** (1) targeting the structural attention bias that suppresses dynamics in autoregressive video diffusion; (2) replacing the static anchor with an adaptive state trained via horizon-weighted DMD; and (3) showing that denoising is itself a recurrence, a hidden latent carried via the KV cache with no external module.

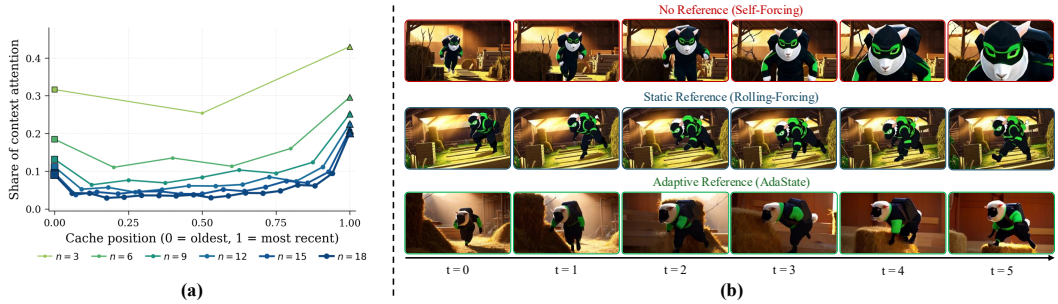


Figure 2: **The anchor-recency structure of streaming video attention.** (a) Off-diagonal attention in Self-Forcing across chunk depths. The anchor at position 0 (squares) and the freshest chunk frame (triangles) consistently dominate; remaining positions receive roughly uniform mass. (b) 5-second generation on the same prompt. Without a persistent reference, coherence degrades over time. A static reference preserves identity but freezes the scene. AdaState maintains identity while the camera moves and the environment evolves.

2 Related Work

Streaming autoregressive video diffusion. Autoregressive video diffusion models generate chunks sequentially with causal attention and KV caching for bounded-cost streaming. Distillation-based training has been the central thread: CausVid [35] introduced asymmetric distillation from bidirectional teachers, Self-Forcing [12] trains on model outputs via DMD to close the train-inference gap, Self-Forcing++ [4] extends this to minute-scale, and Causal Forcing [38] refines the per-frame objective. Rolling Forcing [17] jointly denoises a rolling window, Reward Forcing [18] adds reward-weighted distillation, and MMM [2] couples flow matching on long videos with distribution matching on sliding windows. A common design retains first-frame KV as a static attention anchor, exploiting the attention-sink phenomenon [29]; LongLive [31] makes this explicit by pinning the first frame's KV as a permanent sink. Approaches that update the anchor are content-agnostic: EMA over evicted tokens (Reward Forcing), dual-rate EMA with online RoPE re-indexing (MemRoPE [14]), raw-content replacement on a heuristic schedule (Rolling Sink [16]), spatial-hierarchy compression (PackForcing [20]), and block-relativistic positional encoding (Infinity-RoPE [32]). All either freeze the anchor or update it externally; AdaState updates it through the model's own denoising process, the same computation that produces content.

Persistent state and test-time adaptation. RNNs, LSTMs, and modern selective state-space models [25, 11, 3, 8] maintain hidden states updated at each step by a learned transition function for bounded-cost sequence modeling. Test-Time Training (TTT) [27] reframes recurrence by making the hidden state a model itself, updated via self-supervised learning at each step. Titans [1] introduces a neural long-term memory module with adaptive forgetting; LaCT [37] combines large-chunk TTT with sliding-window attention for video generation; VideoSSM [36] applies SSM-based global context to autoregressive video diffusion. Our work shares the structural property of a persistent hidden variable that is updated at each step and conditions the output without being observed directly. The distinguishing feature is the transition function: rather than a learned gate, recurrence matrix, SSM convolution, or gradient-based update, the state transition is the diffusion model's own multi-step denoising, a pretrained iterative refinement process repurposed as a recurrent update.

Latent reasoning and thinking tokens. A growing line of work augments transformers with latent positions that carry intermediate computation, shaped by task loss rather than explicit supervision: Scratchpads [21] pioneered intermediate computation tokens for language models, Pause tokens [7] showed that empty positions before output improve quality, and [23] demonstrated filler tokens enable hidden computation. Coconut [9] feeds hidden states back as input embeddings in continuous space; CODI [26] aligns latent states with token embeddings via distillation; Huginn [6] uses a looped transformer with recurrent processing. Most recently, [10] showed that latent tokens in diffusion language models, jointly predicted but never decoded, improve reasoning. Our adaptive state applies this principle to video generation: a latent slot processed alongside content, shaped by the generation loss, and serving as a queryable scene reference. The critical difference is persistence — language thinking tokens typically exist within a single forward pass, while our state persists across chunks via

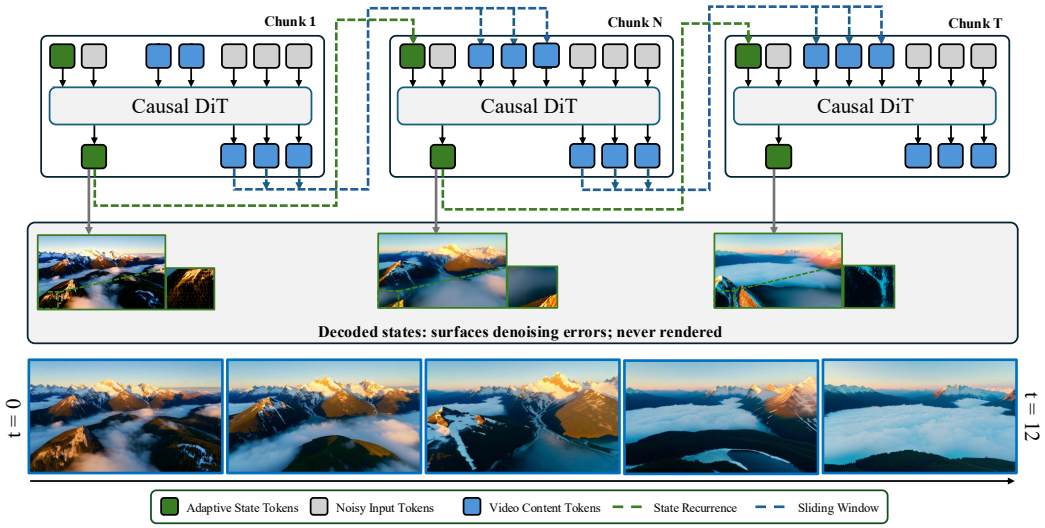


Figure 3: **AdaState Framework.** To denoise chunk 1, the generator takes the first frame of chunk 0 as its reference and the other two frames as recent context. Conditioned on these frames, it jointly denoises an adaptive state (green) and three new content frames. To carry this information into the next chunk, the model writes the clean state keys and values to cache position 0 (green dashed arrows) and retains the new content frames in a sliding window (blue dashed arrows). Each subsequent chunk thus attends to an evolving reference and recent video content. From left to right, its inputs are the cached reference, live state, cached content, and live content (Eq. (1)). The middle row visualizes hidden states sampled across the rollout through diagnostic VAE decodes; only the video content appears in the generated sequence, shown in the blue-bordered strip from t=0 to t=12.

the KV cache and is updated through iterative denoising, functioning as a recurrent hidden variable that carries scene information across generation steps.

3 Method

Preliminaries. We build AdaState on the autoregressive video diffusion framework of Self-Forcing [12], where a student generator G_θ is distilled from a bidirectional teacher via Distribution Matching Distillation (DMD) [34]. Generation proceeds autoregressively in chunks of F latent frames, each denoised from noise through a K-step schedule following the flow matching interpolation $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ [5]. At each chunk n, the generator denoises content $\mathbf{x}_c^{(n)}$ conditioned on a KV cache $\mathbf{h}_n$ of clean key-value pairs from prior chunks. The DMD loss drives this distillation by matching score functions between a frozen teacher S_{real} and a learned critic S_{fake} on the student's noised predictions, with gradient $\nabla_{\text{DMD}} = (S_{\text{fake}}(\hat{\mathbf{v}}_t, t) - S_{\text{real}}(\hat{\mathbf{v}}_t, t))/\gamma$ applied as a pseudo-target [34]. To enable generation beyond a fixed context length, the KV cache operates as a sliding window of size W, where keys are stored without positional encoding and re-encoded at read time with block-relativistic RoPE [32], mapping the visible window to constant relative positions regardless of generation progress. As content exits the window, existing methods compensate by retaining the first frame's clean KV at a fixed sink position as a static scene anchor [31, 17], or by applying EMA over evicted tokens [18]. Following Self-Forcing [12], G_θ is trained by performing the full autoregressive rollout, so the student learns to generate from its own imperfect outputs.

3.1 Context Utilization in Self-Forcing

We probe Self-Forcing's attention over the cached KV window: for each denoising step, we measure post-softmax attention mass per cached K-frame, averaged over heads and renormalized over off-diagonal positions so all cached frames compete for the same mass budget. Figure 2(a) reveals a persistent bimodal structure across all chunk depths: the anchor at position 0 and the freshest chunk-summary frame consistently dominate, while the remaining $\sim$70% of cached positions receive roughly uniform attention. The anchor's absolute share decays with cache size, while its relative

position ranks 2nd-3rd in all cases. Generation is thus driven primarily by what occupies these two positions, not by the full cached history. Figure 2(b) illustrates the consequence: in Self-Forcing, the first frame's KV remains at position 0 throughout generation, and its persistent attention mass constrains the scene from evolving naturally. Methods that pin a static anchor further amplify this, preserving identity but freezing the scene; an adaptive reference at the same position lifts the constraint, maintaining identity while the camera tracks and the environment evolves. Rather than redistributing attention across the cache, we intervene at the position the model already attends to most and replace its frozen content with an evolving, self-generated scene reference.

3.2 Adaptive State

We replace the static first-frame anchor with an *adaptive state* $s \in \mathbb{R}^{F_s \times H \times W \times C}$, a hidden latent that the generator denoises alongside content at every chunk but never renders (see Figure 3). At each chunk n where the adaptive state is active, the *live denoising block* contains $F_s{+}F$ latent frames: F_s state frames and F content frames. The full *attention-visible window* additionally includes F_s cached state frames from the preceding chunk and W_p cached content frames, totaling $2F_s{+}W_p{+}F$ frame-level blocks. With $F_s{=}1$, $W_p{=}3$, and $F{=}3$, the live denoising block contains four latent frames (one state and three content frames), while the full attention-visible window contains eight frame-level blocks (one cached state, one live state, three cached content frames, and three live content frames). Cached keys and values are computed from clean latents at $\sigma{=}0$; live state and content use the current denoising noise level, preserving the backbone's clean-cache/noisy-live convention.

$$\mathbf{h}_n = \big[\ \underbrace{\mathrm{KV}(s_{n-1})}_{F_s}\ ;\ \underbrace{\tilde{s}_n}_{F_s}\ ;\ \underbrace{\mathrm{KV}(\hat{\mathbf{x}}_c^{(\text{past})})}_{W_p \text{ frames}}\ ;\ \underbrace{\mathbf{x}_c^{(n)}}_{F \text{ frames}}\ \big] \tag{1}$$

State positioning. Under causal softmax, attention mass concentrates disproportionately on the earliest positions [29], making position 0 the most influential in the visible window. Existing methods exploit this by placing a frozen first frame there, which provides stable identity grounding but locks the scene reference for the entire generation. We instead place the adaptive state at this position, so that the model's dominant reference point evolves with the scene rather than remaining fixed. This choice also resolves a structural discontinuity introduced by block-relativistic RoPE [32]: while the sliding window is re-indexed at each chunk so that content positions are always relative, a static anchor at position 0 remains an absolute fixed point whose content never changes. An adaptive state that is regenerated each chunk makes position 0 consistent with the relative-time semantics of the rest of the window.

Recurrence. Both state and content start from independent Gaussian noise and are denoised jointly:

$$[s_n,\, \hat{\mathbf{x}}_c^{(n)}] = G_\theta([\boldsymbol{\epsilon}_s,\, \boldsymbol{\epsilon}_c]\,;\, \mathbf{h}_n)\,, \quad \boldsymbol{\epsilon}_s, \boldsymbol{\epsilon}_c \sim \mathcal{N}(0, I) \tag{2}$$

where $\hat{\mathbf{x}}_c^{(n)}$ and s_n are the clean content and state predictions, and $\mathbf{h}_n$ is the cache from Eq. (1). After denoising, the two predictions follow different paths: the content is decoded into video and its clean KV enters the sliding window for short-term context, while the state is never decoded but instead overwrites position 0 to serve as the scene reference for the next chunk:

$$\mathbf{h}_{n+1}^{(s)} = \mathrm{KV}(s_n), \qquad \mathrm{KV}(\hat{\mathbf{x}}_c^{(\text{past})}) = \mathrm{KV}(\hat{\mathbf{x}}_c^{(j-W_p+1:j)}) \tag{3}$$

Here j is the final frame index of chunk n, and the cache retains the most recent W_p content frames.

This cache update creates the recurrence: when chunk $n{+}1$ begins, the generator denoises new content and state from noise, but the cache now carries s_n's clean KV at position 0. Because the new state s_{n+1} attends to this cached representation alongside the current content, the generator G_θ serves as the state transition function and the KV cache as the carrier. This parallels the recurrence in state models such as RNNs [3] and LSTMs [11]: a hidden variable, updated at each step by the model's own computation, that conditions the output without being observed directly.

Information flow. During each denoising pass, queries from the live tokens attend to the full visible window. This creates two complementary information flows: content queries read the cached state for scene context that has been evicted from the sliding window, while the live state's queries read current content to absorb the evolving scene. Because the live state starts from pure noise with no structural prior encoding the previous chunk, identity and scene context must be actively reconstructed through

attention to the cached KV at position 0, rather than passively copied from the input. Chunk 0 generates only content, with the first frame's keys and values serving as the initial reference at cache position 0. Starting at chunk 1, the generator jointly denoises the adaptive state and new content, then writes the state's clean keys and values to position 0 for the next chunk.

$$\mathrm{Attn}([\mathbf{q}_{\tilde{s}_n};\, \mathbf{q}_c^{\text{live}}],\ [\mathbf{k}_{s_{n-1}};\, \mathbf{k}_{\tilde{s}_n};\, \mathbf{k}_c^{\text{cache}};\, \mathbf{k}_c^{\text{live}}],\ [\mathbf{v}_{s_{n-1}};\, \mathbf{v}_{\tilde{s}_n};\, \mathbf{v}_c^{\text{cache}};\, \mathbf{v}_c^{\text{live}}]) \tag{4}$$

Equation 4 makes explicit that the state participates in the same attention mechanism as content, no separate module or gating is required. The full recurrence thus consists of three standard operations: joint denoising (Eq. 2), cache update (Eq. 3), and attention (Eq. 4), all already present in the pretrained backbone. The state's representation is shaped entirely by the generation loss propagated through content attention; no auxiliary objective or supervision is needed to teach the model what to store.

3.3 Horizon-Weighted Training

In autoregressive generation, errors propagate and amplify through the chunk chain, so the later training frames experience the most accumulated drift and preview beyond-horizon conditions. Under a uniform loss, these critical late frames are underweighted, early frames, being well-conditioned, dominate the mean loss and absorb optimizer capacity. The frames that matter most for generalization receive the least optimization pressure. We address this by weighting the DMD loss per frame with a linear ramp that increases with frame index:

$$\mathcal{L} = \frac{1}{2}\sum_{i=0}^{N-1} w_i \left\| \hat{\mathbf{x}}_i - \left(\hat{\mathbf{x}}_i - \frac{S_{\text{fake}}(\hat{\mathbf{x}}_{i,t}, t) - S_{\text{real}}(\hat{\mathbf{x}}_{i,t}, t)}{\gamma} \right)_{\text{sg}} \right\|^2, \quad w_i = 1 + \alpha \cdot \frac{i}{N-1} \tag{5}$$

where $\hat{\mathbf{x}}_i$ is the clean prediction of frame i, $\hat{\mathbf{x}}_{i,t} = (1-t)\hat{\mathbf{x}}_i + t\boldsymbol{\epsilon}$ is its noised version at level t, N is the number of frames in the rollout, and α controls the ramp slope, redirecting the optimizer toward the later frames where drift accumulates. This weighting is particularly important for the adaptive state, since the late frames are precisely the ones where the original scene content has exited the sliding window and the cached state at position 0 becomes the primary scene reference. No separate loss is applied to the state; instead, gradient reaches it entirely through the attention that content frames pay to it. Because the horizon weighting concentrates the loss on the frames whose quality depends most on the state's contribution, the training signal naturally shapes the state to provide useful scene context where it matters most. The recurrence is present during both training and inference, but cached states are detached at chunk boundaries: training does not perform backpropagation through time (BPTT) or directly correct a previous cached state. Instead, within-chunk gradients shape the live state through its interaction with content while the generator learns under model-generated recurrent context. We therefore view AdaState as a recurrent inference state trained through autoregressive rollout, rather than BPTT-trained memory; state interventions in Appendix D.2 test its downstream influence.

4 Experiments

We build on Wan2.1-T2V-1.3B [28], distilled into a causal autoregressive generator via Self-Forcing [12] with DMD loss against a Wan2.1-T2V-14B teacher. Each chunk denoises $F{=}3$ latent frames through a 4-step schedule alongside $F_s{=}1$ adaptive state frame, with $W_p{=}3$ cached content frames from prior chunks providing local context. Starting from the Self-Forcing checkpoint, we fine-tune on 21-frame rollouts (seven chunks) for 1000 iterations with horizon-weighted DMD using their training prompts, $\alpha{=}2$ for within-horizon evaluation, $\alpha{=}4$ for long-horizon generation, at learning rate $2{\times}10^{-6}$ and effective batch size 4 on two H200 GPUs. Evaluation uses VBench [13] at 5 seconds (21 latent frames, within training horizon) and 30 seconds (120 latent frames, 6$\times$ training horizon), and VisionReward [30] at 5 seconds. After completing the autoregressive rollout, we decode the generated content latents using Wan-VAE, which processes the sequence in temporal chunks while carrying forward causal-convolution feature caches. The adaptive-state latents are not decoded. Appendix E provides further details on decoding and computational cost.

We compare against methods spanning anchor mechanisms for streaming generation: no persistent anchor (Self-Forcing [12], CausVid [35], Causal Forcing [38]), static anchor (LongLive [31], Rolling Forcing [17], Infinity-RoPE [32]), EMA and positional updates (Reward Forcing [18], Mem-RoPE [14]), and a heuristic anchor replacement (Rolling Sink [16]). The non-autoregressive Wan 2.1-1.3B serves as a non-autoregressive quality reference.

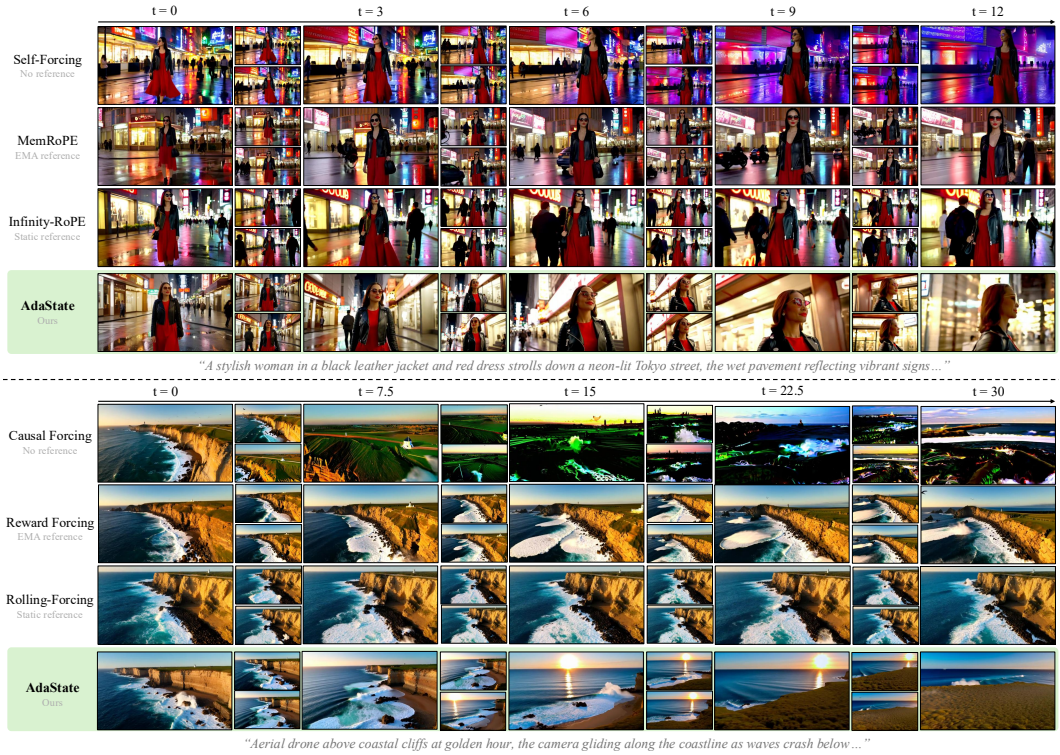


Figure 4: **Qualitative comparison across anchor categories.** Top block: 12-second generation sampled at 3-second intervals. Bottom block: 30-second generation sampled at 7.5-second intervals. Each block pairs AdaState against one exemplar per baseline category (no reference, EMA reference, static reference); the six exemplars across the two blocks cover all baseline groups. Methods without a persistent anchor accumulate color drift (Self-Forcing, top) or collapse to artifacts at long horizons (Causal Forcing, bottom). EMA and static references prevent collapse but freeze the scene: MemRoPE, Infinity-RoPE, Reward Forcing, and Rolling-Forcing each reproduce a near-identical composition across the entire rollout. AdaState alone preserves identity while producing natural scene evolution.

4.1 Qualitative Results

Figure 4 pairs AdaState against one exemplar per baseline category at two horizons. The top block presents a 12-second portrait rollout. Self-Forcing, trained on 5-second rollouts, accumulates visual artifacts beyond its training horizon: color drift emerges early and compounds through the rest of the sequence. MemRoPE and Infinity-RoPE prevent drift but lock the composition: their rollouts reproduce a near-identical scene across all keyframes, with no progression of camera or scene. The bottom block extends to 30 seconds on a drone shot of a coastline at golden hour. Causal Forcing, similarly trained on 5-second rollouts, collapses to visual artifacts well beyond its training horizon, no longer corresponding to the prompt. Rolling-Forcing and Reward Forcing prevent the collapse but freeze the scene similarly, with their static/EMA references. AdaState alone produces both temporal stability and natural progression: across the 12-second portrait shot, the scene evolves continuously with camera motion and subject action; across the 30-second coastal drone shot, the camera glides along the shoreline, revealing new terrain in continuous golden hour light. We provide further qualitative examples in the supplementary material.

4.2 Quantitative Results

We frame streaming methods by their anchor mechanism: no persistent anchor, static reference, EMA-based or heuristic update, and adaptive. These categories exhibit a consistency-dynamics tradeoff: methods that preserve a stable reference tend to suppress scene evolution, while those that allow greater motion risk accumulated drift. AdaState improves this tradeoff by learning an

Table 1: **Evaluation in 5s and 30s rollouts.** 5 seconds (21 frames) is within all methods' training horizons; 30 seconds (120 frames) is six times the training length. VisionReward (VR) is reported at 5 seconds only. Best in **bold**, second best underlined.

	5 seconds (within training horizon)						30 seconds (6× training horizon)				
Method	Subj.↑	Dyn.↑	Aesth.↑	Avg.↑	Consist.↑	$VR_{(\times 10)}$↑	Subj.↑	Dyn.↑	Aesth.↑	Avg.↑	Consist.↑
Non-autoregressive											
Wan 2.1-1.3B	0.970	0.609	0.593	0.823	0.253	0.664	–	–	–	–	–
No persistent anchor											
Self-Forcing	0.966	0.635	0.619	0.832	0.249	0.806	0.972	0.392	0.570	0.794	0.224
CausVid	0.979	0.547	0.606	0.819	0.238	0.620	0.986	0.096	0.626	0.761	0.213
Causal Forcing	0.974	0.747	0.628	0.851	0.246	0.774	0.944	**0.933**	0.523	0.843	0.194
Static anchor											
LongLive	0.976	0.393	**0.635**	0.806	0.253	0.840	0.983	0.373	0.625	0.801	0.255
Rolling Forcing	**0.983**	0.367	0.619	0.804	0.251	0.864	**0.991**	0.042	**0.648**	0.769	0.254
Infinity-RoPE	0.978	0.628	0.605	0.832	0.245	0.820	0.986	0.267	0.638	0.788	0.237
EMA / positional / heuristic											
Reward Forcing	0.975	0.643	0.622	0.838	0.248	0.791	0.979	0.607	0.628	0.837	0.252
MemRoPE	0.976	0.622	0.622	0.836	0.253	0.798	0.974	0.718	0.601	0.848	0.249
Rolling Sink	0.979	0.508	0.627	0.822	**0.255**	0.812	0.977	0.402	0.621	0.807	0.252
Adaptive anchor (ours)											
AdaState	0.961	**0.828**	0.630	**0.860**	0.254	**0.868**	0.959	0.922	0.596	**0.865**	**0.259**

adaptive state that evolves with the generated content. We tested this behavior at two horizons: within (5 seconds) and beyond training (30 seconds, six times the training horizon). Evaluation uses MovieGenBench Extended [24] prompts spanning static and dynamic scenes at 5 seconds, and aerial or drone-style moving-camera prompts at 30 seconds. Appendix C provides representative samples from these evaluation lists. To assess temporal degradation within the same rollouts, Appendix D.1 reports interval-level evaluations through 45 seconds. We report VBench [13] and VisionReward [30] in Table 1, with the consistency-dynamics tradeoff visualized at long horizon in Figure 5. Within the training horizon, baselines populate the predicted diagonal by category: static-anchor methods (LongLive, Rolling Forcing) cluster at the high-consistency, low-dynamics end; anchor-free methods (Self-Forcing, Causal Forcing) sit at the opposite end, recovering dynamics at the cost of consistency; EMA and heuristic anchor-update methods land along the same line without escaping it, since content-agnostic updates cannot adapt the reference to evolving content. AdaState is the only method off the diagonal: it posts the highest VBench total, dynamic degree, and VisionReward, with subject consistency just below the static-anchor cluster, the price of replacing a frozen reference with an evolving one.

We then push to 30 seconds to stress-test whether the tradeoff holds beyond training horizon. The categorical structure persists and sharpens (Figure 5): six of nine baselines fall below 0.5 dynamic degree with Rolling Forcing and CausVid effectively frozen, and among methods that maintain motion, Causal Forcing pays by drifting off the prompt, its subject consistency dropping noticeably from the 5-second value. AdaState combines high dynamics with higher subject consistency than Causal Forcing, and achieves the highest VBench average and text-video alignment at this horizon. Its dynamic degree is close to Causal Forcing, while static-anchor methods obtain higher scores on some consistency measures at substantially lower dynamics. Together, these results demonstrate AdaState's improved balance between scene evolution and consistency. Fixed-α controls and within-rollout analyses below further characterize the contributions of the adaptive state and horizon weighting.

To confirm the perceptual ranking, we conduct a user study with 40 raters with Prolific platform[1]. Each rater views videos from AdaState and four representative baselines: Self-Forcing and Causal Forcing (no-reference), Infinity-RoPE (static-reference), and MemRoPE (EMA-reference), on 20 videos spanning both horizons, scoring each on coherent progression and prompt adherence. The ratings mirror the categorical structure (Figure 6): no-reference methods score lowest, static and EMA-reference methods occupy the middle, and AdaState leads on both dimensions. The largest gap falls against the no-reference category, where beyond-horizon artifacts penalize both ratings; the ordering matches the failure modes from Figure 4, with locked or artifact-laden videos rated below those that progress coherently. We provide further details on our user study in the supplementary material.

[1] `https://www.prolific.com/`

Table 2: **Architecture and training ablations.** (a) State size, content window, weighting, and static-sink controls at 5s; Avg. is the mean of the six listed metrics. (b) No-state controls and fixed-α AdaState models evaluated at both horizons after 1000 fine-tuning iterations; the original Self-Forcing model is a separate baseline. AdaState uses one state frame (F_s=1), three cached content frames (W_p=3), and three live content frames (F=3), giving an eight-frame attention-visible window including the cached and live states. α=0 denotes uniform weighting; higher scores are better. Best scores within each panel are in **bold**, including ties.

(a) Architecture and anchor configuration (5 seconds)

Configuration	Sink	F_s	W_p+F	α	Subj.↑	BG↑	Smooth↑	Dyn.↑	Aesth.↑	Img.↑	Avg.↑
State, F_s=3	–	3	9	0	0.971	0.964	0.984	0.602	0.623	**0.721**	0.811
State, F_s=1	–	1	9	0	0.962	0.959	0.980	0.581	**0.631**	0.700	0.802
State, F_s=1	–	1	6	0	0.965	0.956	0.981	0.734	0.627	0.707	0.829
State, F_s=1, α=2	–	1	6	2	0.961	0.953	0.980	**0.828**	0.630	0.702	**0.842**
State, F_s=1, α=4	–	1	6	4	0.971	0.961	0.984	0.688	0.628	0.712	0.824
Sink + State, α=4	✓	1	6	4	**0.977**	**0.966**	**0.986**	0.570	0.629	0.703	0.805

(b) State and horizon-weighting controls

		5 seconds				30 seconds			
Configuration	α	Subj.↑	Dyn.↑	Aesth.↑	Img.↑	Subj.↑	Dyn.↑	Aesth.↑	Img.↑
Self-Forcing	–	0.966	0.635	0.619	0.697	0.972	0.392	0.570	0.686
No-state	2	0.971	0.484	0.604	0.699	0.990	0.178	**0.610**	0.676
No-state	4	**0.973**	0.492	0.609	0.697	**0.992**	0.078	**0.610**	0.699
AdaState	0	0.965	0.734	0.627	0.707	0.967	0.787	0.609	**0.705**
AdaState	2	0.961	**0.828**	**0.630**	0.702	0.958	0.893	0.592	0.703
AdaState	4	0.971	0.688	0.628	**0.712**	0.959	**0.922**	0.596	0.685

4.3 Ablation Study

We ablate the main design choices in AdaState to examine how the state configuration, training objective, and anchor mechanism contribute to generation quality (Table 2). These experiments also distinguish the contribution of the adaptive state from that of horizon weighting.

State size and content window. We first examine how the state size and amount of recent context affect generation. With the state size held fixed, reducing the content window produces stronger dynamics. Adding more state frames to the wider window does not offer the same benefit, suggesting that capacity alone does not determine how effectively the state guides generation. A single recurrent state paired with a short content window provides an effective balance: recent frames support local continuity, while the state supplies an evolving scene reference.

Horizon weighting. We next examine how emphasizing later frames affects generation. Moderate weighting favors within-horizon dynamics, while stronger weighting improves subject consistency and imaging quality. To distinguish these effects from the adaptive state's contribution, Table 2(b) evaluates models trained with fixed horizon weights at both horizons, alongside no-state controls trained with the same fine-tuning budget. AdaState's dynamics gains persist under uniform weighting and with the same model at both horizons, whereas weighting alone does not yield the same improvement. The adaptive state provides an evolving reference, while horizon weighting emphasizes later frames during training.

Retaining a static anchor. Finally, we test whether a static anchor can complement the adaptive state. Keeping both reduces dynamics, suggesting that the fixed reference continues to constrain scene progression even when an evolving alternative is available. This supports replacing the static anchor so that the adaptive state can guide generation as the scene develops.

Further analyses of temporal stability and coherence, cached-state interventions, and computational cost and initialization are provided in Appendices D.1, D.2, and E, respectively.

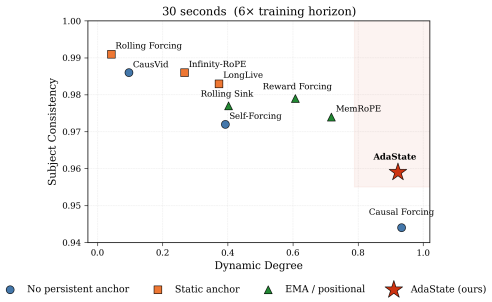


Figure 5: **Subject consistency vs. dynamic degree at 30 seconds.** The dynamics distribution becomes bimodal: most baselines collapse to the left as motion stops, while AdaState alone occupies the shaded upper-right region where high dynamics and high consistency coexist.

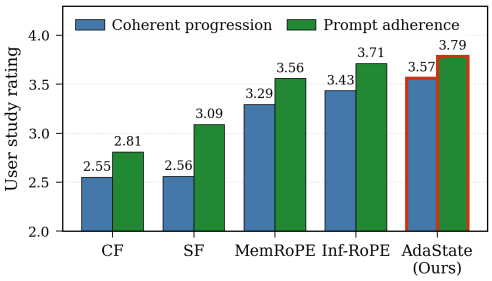


Figure 6: **User study (5-point Likert, N=40 raters).** Methods are ordered by coherent-progression score. AdaState gets the highest ratings on both coherent progression and prompt following.

5 Limitations

Our current implementation uses a single latent state frame (F_s=1), which provides an effective balance between dynamics and consistency in the evaluated settings. Increasing the number of state slots does not improve dynamics in our ablations, suggesting that richer state representations may benefit from structured roles rather than additional capacity alone. The interval-level evaluation through 45 seconds in Appendix D.1 (Table 8) shows sustained scene evolution alongside gradual changes in local detail. Small non-rigid subjects, particularly around occlusion or re-entry, remain a challenging setting for preserving fine appearance details over extended rollouts. Appendix D.1 discusses representative cases and their relation to the interval-level results.

These observations motivate exploring structured external memory to complement the adaptive state in longer rollouts and scenes with multiple actors or concurrent events. Such mechanisms could retain fine-grained information beyond the local context while allowing the scene reference to evolve. More broadly, the attention-sink pattern underlying AdaState occurs across transformer architectures [29], making adaptive recurrent states a promising direction for other autoregressive generation domains. Our video experiments provide an initial setting for studying this formulation, with broader applications offering opportunities for future work.

6 Conclusion

We propose AdaState, a recurrent formulation for streaming video generation that addresses the tension between maintaining a stable scene reference and allowing the scene to evolve. Rather than anchoring generation to a frozen first frame, AdaState lets the model regenerate its own hidden scene reference through the same denoising process that produces video content. Carried across chunks through the KV cache and trained with horizon-weighted DMD, this adaptive state supports scene progression without requiring an external memory module or changes to the backbone architecture. Our experiments show that this design improves the balance between consistency and dynamics within and beyond the training horizon, with gains that persist under fixed horizon weighting and are supported by human judgments of coherent progression and prompt adherence. AdaState thus establishes denoising as a mechanism for recurrent state evolution, suggesting a broader direction for autoregressive generation in which the model learns not only what to generate next, but also how to update the context that guides its continuation.

Acknowledgements

This research is supported by the National Science Foundation under Grant No. 2543524. Yusuf Dalva is supported by a grant from the Amazon-Virginia Tech Initiative for Efficient and Robust Machine Learning. We also gratefully acknowledge fal.ai for providing compute support.

References

[1] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[2] Shengqu Cai, Weili Nie, Chao Liu, Julius Berner, Lvmin Zhang, Nanye Ma, Hansheng Chen, Maneesh Agrawala, Leonidas Guibas, Gordon Wetzstein, et al. Mode seeking meets mean seeking for fast long video generation. *arXiv preprint arXiv:2602.24289*, 2026.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[4] 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 *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=DzvPiqh23f`.

[5] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=FPnUhsQJ5B`.

[6] Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=S3GhJooWIC`.

[7] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=ph04CRkPdC`.

[8] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[9] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=Itxz7S4Ip3`.

[10] Andre He, Sean Welleck, and Daniel Fried. Reasoning with latent tokens in diffusion language models. *arXiv preprint arXiv:2602.03769*, 2026.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[12] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.

[13] 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*, 2025. doi: 10.1109/TPAMI.2025.3633890.

[14] Youngrae Kim, Qixin Hu, C-C Jay Kuo, and Peter A Beerel. Memrope: Training-free infinite video generation via evolving memory tokens. *arXiv preprint arXiv:2603.12513*, 2026.

[15] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *European conference on computer vision*, pages 179–195. Springer, 2018.

[16] 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*, 2026.

[17] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025.

[18] 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. *arXiv preprint arXiv:2512.04678*, 2025.

[19] Ge Ya Luo, Gian M Favero, Zhi Hao Luo, Alexia Jolicoeur-Martineau, and Christopher Pal. Beyond fvd: An enhanced evaluation metrics for video generation distribution quality. In *International Conference on Learning Representations*, volume 2025, pages 66218–66252, 2025.

[20] Xiaofeng Mao, Shaohao Rui, Kaining Ying, Bo Zheng, Chuanhao Li, Mingmin Chi, and Kaipeng Zhang. Packforcing: Short video training suffices for long video sampling and long context inference. *arXiv preprint arXiv:2603.25730*, 2026.

[21] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*, 2022. URL `https://openreview.net/forum?id=HBlx2idbkbq`.

[22] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[23] Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=NikbrdtYvG`.

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

[25] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, 1985.

[26] Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 677–693, 2025.

[27] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=wXfuOj9C7L`.

[28] 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 preprint arXiv:2503.20314*, 2025.

[29] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=NG7sS51zVF`.

[30] Jiazheng Xu, Yu Huang, Jiale Cheng, Yuanming Yang, Jiajun Xu, Yuan Wang, Wenbo Duan, Shen Yang, Qunlin Jin, Shurun Li, et al. Visionreward: Fine-grained multi-dimensional human preference learning for image and video generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 11269–11277, 2026.

[31] Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Ying-Cong Chen, Yao Lu, Song Han, and Yukang Chen. Longlive: Real-time interactive long video generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=nCAODkpsPJ.

[32] 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. *arXiv preprint arXiv:2511.20649*, 2025.

[33] 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. *arXiv preprint arXiv:2512.05081*, 2025.

[34] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024.

[35] 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 *CVPR*, 2025.

[36] Yifei Yu, Xiaoshan Wu, Xinting Hu, Tao Hu, Yangtian Sun, Xiaoyang Lyu, Bo Wang, Lin Ma, Yuewen Ma, Zhongrui Wang, et al. Videossm: Autoregressive long video generation with hybrid state-space memory. *arXiv preprint arXiv:2512.04519*, 2025.

[37] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

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

Table of Contents

A Implementation Details **15**

B User Study Details **16**

C Evaluation Details **16**

D Supplementary Experiments **17**

D.1 Stability and Coherence over Time 17

D.2 Cached-State Interventions 20

E Compute, Decoding, and Initialization **20**

F Supplementary Video Results **21**

G Attention Analysis Details **22**

H Limitations and Future Work **23**

I Broader Impact and Ethics **23**

A Implementation Details

This section provides the full training algorithm and hyperparameter configuration for AdaState. We build on the Self-Forcing codebase and checkpoint, modifying only the cache management (state slot at position 0, content sliding window) and the loss weighting (horizon ramp). No architectural changes are made to the backbone DiT; the adaptive state occupies the same latent space as content frames and is processed by the same layers.

Training algorithm. Algorithm 1 summarizes the AdaState training procedure. Each iteration performs a full autoregressive rollout of N_c chunks. From chunk 1 onward, the generator jointly denoises F_s state frames and F content frames from independent Gaussian noise, conditioned on the KV cache. After denoising, the content's clean KV enters the sliding window while the state's clean KV overwrites position 0. The horizon-weighted DMD loss (Eq. 5 in the main paper) is applied per frame, and the cross-chunk state recurrence is detached at chunk boundaries.

Algorithm 1 AdaState Training

Require: Generator G_θ, critic S_{fake}, frozen teacher S_{real}, chunk count N_c, chunk size F, state size F_s, cached content frames W_p, denoising steps K, horizon weight α, learning rate η

while training **do**
  Sample prompt p
  Generate $\mathbf{x}_c^{(0)}$ with G_θ conditioned on p ▷ chunk 0: content only
  $\mathbf{h} \leftarrow \mathrm{KV}(\mathbf{x}_c^{(0)})$ ▷ init cache from chunk 0
  for $n = 1, \ldots, N_c - 1$ **do**
    Sample $\boldsymbol{\epsilon}_s, \boldsymbol{\epsilon}_c \sim \mathcal{N}(0, I)$
    $[s_n,\, \hat{\mathbf{x}}_c^{(n)}] \leftarrow G_\theta([\boldsymbol{\epsilon}_s, \boldsymbol{\epsilon}_c]\,;\, \mathbf{h})$ ▷ first adaptive state at $n = 1$
    $w_i = 1 + \alpha \cdot i \,/\, (N{-}1)$ for global frame index i ▷ horizon weights
    $\hat{\mathbf{x}}_{i,t} = (1{-}t)\,\hat{\mathbf{x}}_i + t\,\boldsymbol{\epsilon}, \quad t \sim \mathcal{U}(0,1)$ ▷ noise predictions
    $\nabla_{\text{DMD}} = (S_{\text{fake}}(\hat{\mathbf{x}}_{i,t}, t) - S_{\text{real}}(\hat{\mathbf{x}}_{i,t}, t)) \,/\, \gamma$ ▷ DMD gradient
    $\mathcal{L}_n = \frac{1}{2} \sum_i w_i \left\| \hat{\mathbf{x}}_i - (\hat{\mathbf{x}}_i - \nabla_{\text{DMD}})_{\text{sg}} \right\|^2$ ▷ weighted loss
    Cache the latest W_p content frames ▷ slide content window
    $\mathbf{h}^{(s)} \leftarrow \mathrm{KV}(\mathrm{stopgrad}(s_n))$ ▷ detach state across chunks
  end for
  $\mathcal{L} \leftarrow \sum_n \mathcal{L}_n$
  $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}$ ▷ generator update
  Update S_{fake} on $\{(\hat{\mathbf{x}}_{i,t}, t)\}$ ▷ critic update
end while

Hyperparameters. Table 3 lists the full training configuration. We fine-tune from the Self-Forcing checkpoint for 1000 iterations using AdamW (β_1=0.9, β_2=0.999, weight decay 0.01) on two H200 GPUs. The only hyperparameter that differs between models is the horizon weight α: we use α=2 for within-horizon evaluation and α=4 for long-horizon generation, where concentrating gradient on late frames directly benefits the extended rollout. EMA averaging begins at iteration 200 with decay 0.99.

Table 3: Training hyperparameters.

Parameter	Value
Generator / critic learning rate	2×10^{-6} / 4×10^{-7}
Optimizer / effective batch size	AdamW / 4
Training hardware / iterations	2×H200 / 1000
Content / state frames per chunk	F=3 / F_s=1
Cached content / denoising steps	W_p=3 / K=4
Training rollout	21 latent frames / 7 chunks
Horizon weight	α=2 (5 s), α=4 (30 s)

B User Study Details

We conduct a user study to evaluate whether human perception aligns with the categorical structure observed in automated metrics. The study compares AdaState against four representative baselines spanning the three baseline categories: Self-Forcing [12] and Causal Forcing [38] (no persistent anchor), Infinity-RoPE [32] (static anchor), and MemRoPE [14] (EMA-based anchor).

We recruit 40 raters through the Prolific platform. Each rater evaluates 20 distinct videos, presented in randomized order with method identity hidden. The prompt set includes 10 within-horizon (5-second) and 10 beyond-horizon (30-second) generations to test whether perceptual quality degrades with rollout length. Videos are hosted as unlisted uploads (anonymous) and embedded in a Microsoft Forms questionnaire; raters are instructed to watch each video in full before responding. Figure 7 shows the evaluation interface.

Rating dimensions. Each video is scored on two dimensions using a 5-point Likert scale (1 = Poor, 5 = Very Good):

- **Coherent progression:** "How coherently the video progresses over time? Please evaluate in terms of camera motion, subject action, or scene development."
- **Prompt adherence:** "How well does the video adhere to the text prompt? Please think about whether the described scene, subject, and motion are realized in the video."

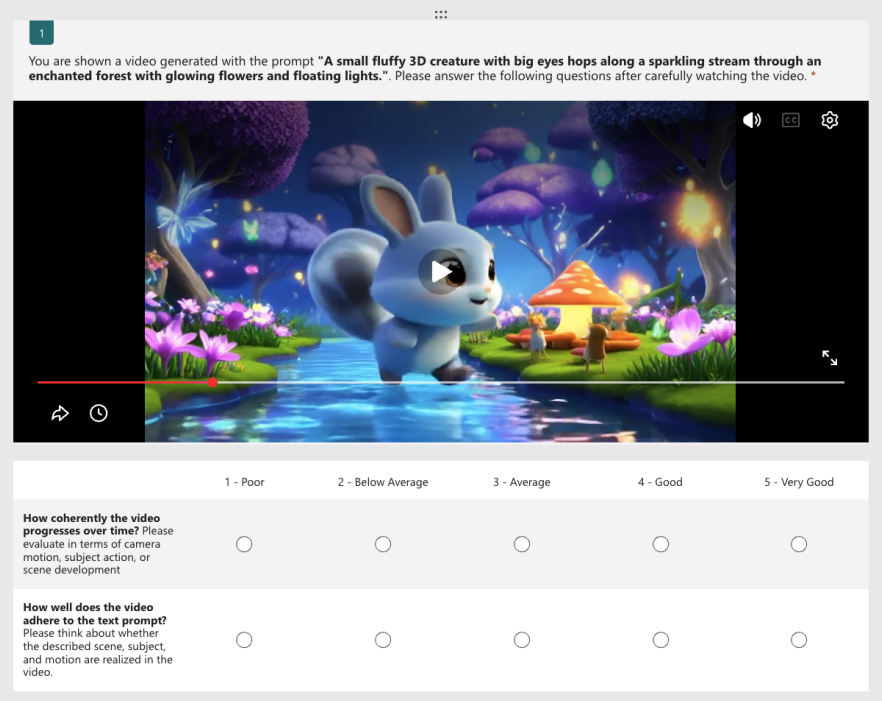


Figure 7: **User study evaluation interface.** Each rater views a video generated from a given prompt and scores it on two dimensions using a 5-point Likert scale. Method identity is hidden; video order is randomized. Identifying information has been redacted for anonymity.

C Evaluation Details

Detailed quantitative results. Tables 4 and 5 provide the full metric breakdown at 5 and 30 seconds, extending Table 1 of the main paper. We report seven VBench quality dimensions, their average, and overall consistency for text-video alignment, together with VisionReward at 5 seconds. Dynamic degree is a binary per-video measure indicating whether meaningful motion is detected, averaged

across evaluation prompts. Its relatively large standard deviation reflects variation between videos classified as dynamic or static, rather than the magnitude of motion within each video. The detailed results further illustrate the balance between consistency and dynamics: static-anchor methods generally obtain stronger consistency scores, while methods that allow greater scene evolution achieve higher dynamics. AdaState achieves the highest VBench average at both horizons, combining strong dynamics with competitive visual quality and text-video alignment.

Table 4: Full VBench breakdown at 5 seconds (21 frames). VisionReward (VR) is reported $\times 10$. Best in **bold**, second best underlined.

Method	Subj.↑	BG↑	Smooth↑	Dyn.↑	Aesth.↑	Img.↑	Flicker↑	Avg.↑	Consist.↑	VR$_{(\times 10)}$↑
Non-autoregressive										
Wan 2.1-1.3B	0.970 ± .03	0.965 ± .01	0.986 ± .01	0.609 ± .43	0.593 ± .09	0.661 ± .10	0.976 ± .02	0.823	0.253 ± .05	0.664 ± .07
No persistent anchor										
Self-Forcing	0.966 ± .03	0.952 ± .02	0.983 ± .01	0.635 ± .44	0.619 ± .09	0.697 ± .07	0.968 ± .03	0.832	0.249 ± .05	0.806 ± .06
CausVid	0.979 ± .02	0.967 ± .01	0.980 ± .01	0.547 ± .45	0.606 ± .10	0.689 ± .08	0.968 ± .02	0.819	0.238 ± .05	0.620 ± .06
Causal Forcing	0.974 ± .02	0.965 ± .01	0.976 ± .02	0.747 ± .39	0.628 ± .09	0.710 ± .07	0.955 ± .03	0.851	0.246 ± .05	0.774 ± .06
Static anchor										
LongLive	0.976 ± .02	0.967 ± .02	0.989 ± .01	0.393 ± .44	0.635 ± .09	0.700 ± .07	**0.998** ± .02	0.806	0.253 ± .05	0.840 ± .06
Rolling Forcing	0.983 ± .01	0.971 ± .01	0.988 ± .01	0.367 ± .43	0.619 ± .09	0.721 ± .05	0.978 ± .01	0.804	0.251 ± .05	0.864 ± .09
Infinity-RoPE	0.978 ± .02	0.968 ± .01	0.986 ± .01	0.628 ± .43	0.605 ± .09	0.683 ± .08	0.975 ± .02	0.832	0.245 ± .05	0.820 ± .06
EMA / positional / heuristic										
Reward Forcing	0.975 ± .02	0.967 ± .01	0.984 ± .01	0.643 ± .46	0.622 ± .09	0.704 ± .07	0.969 ± .02	0.838	0.248 ± .05	0.791 ± .07
MemRoPE	0.976 ± .02	0.967 ± .01	0.983 ± .01	0.622 ± .43	0.622 ± .09	0.712 ± .07	0.968 ± .01	0.836	0.253 ± .05	0.798 ± .06
Rolling Sink	0.979 ± .02	0.970 ± .01	0.986 ± .01	0.508 ± .44	0.627 ± .09	0.708 ± .07	0.975 ± .01	0.822	**0.255** ± .03	0.812 ± .06
Adaptive anchor (ours)										
AdaState	0.961 ± .03	0.953 ± .02	0.980 ± .02	**0.828** ± .39	**0.630** ± .09	0.702 ± .07	0.963 ± .02	**0.860**	0.254 ± .05	**0.868** ± .05

Table 5: Full VBench breakdown at 30 seconds (120 frames, 6× training horizon). Best in **bold**, second best underlined.

Method	Subj.↑	BG↑	Smooth↑	Dyn.↑	Aesth.↑	Img.↑	Flicker↑	Avg.↑	Consist.↑
No persistent anchor									
Self-Forcing	0.972 ± .01	0.962 ± .01	0.991 ± .01	0.392 ± .31	0.570 ± .05	0.686 ± .08	0.983 ± .01	0.794	0.224 ± .05
CausVid	0.986 ± .00	0.972 ± .01	0.991 ± .00	0.096 ± .12	0.626 ± .10	0.666 ± .10	0.987 ± .01	0.761	0.213 ± .04
Causal Forcing	0.944 ± .01	0.949 ± .01	0.970 ± .02	0.933 ± .11	0.523 ± .05	0.637 ± .09	0.944 ± .03	0.843	0.194 ± .03
Static anchor									
LongLive	0.983 ± .01	0.967 ± .01	0.990 ± .00	0.373 ± .34	0.625 ± .04	0.692 ± .07	0.978 ± .01	0.801	0.255 ± .03
Rolling Forcing	**0.991** ± .00	**0.978** ± .01	**0.992** ± .00	0.042 ± .15	0.648 ± .04	**0.747** ± .06	**0.986** ± .01	0.769	0.254 ± .03
Infinity-RoPE	0.986 ± .01	0.972 ± .01	0.991 ± .01	0.267 ± .33	0.638 ± .05	0.680 ± .11	0.983 ± .01	0.788	0.237 ± .03
EMA / positional / heuristic									
Reward Forcing	0.979 ± .01	0.967 ± .01	0.989 ± .01	0.607 ± .40	0.628 ± .05	0.720 ± .08	0.970 ± .01	0.837	0.252 ± .03
MemRoPE	0.974 ± .01	0.962 ± .01	0.990 ± .00	0.718 ± .33	0.601 ± .04	0.717 ± .08	0.973 ± .01	0.848	0.249 ± .03
Rolling Sink	0.977 ± .01	0.965 ± .01	0.990 ± .01	0.402 ± .37	0.621 ± .04	0.711 ± .08	0.980 ± .01	0.807	0.252 ± .03
Adaptive anchor (ours)									
AdaState	0.959 ± .01	0.951 ± .01	0.980 ± .01	**0.922** ± .10	0.596 ± .04	0.685 ± .09	0.962 ± .02	**0.865**	**0.259** ± .03

Evaluation prompts. For 5-second generation, we use MovieGenBench [24] prompts from the extended prompt list released with Self-Forcing [12]. These prompts cover static and dynamic scenes across diverse subjects and visual styles. For 30-second generation, we use prompts describing sustained camera motion through varied environments, including landscapes, cityscapes, and underwater scenes, to evaluate continued scene progression beyond the training horizon. Tables 6 and 7 present representative samples from the prompt lists used for evaluation, shortened for readability.

D Supplementary Experiments

D.1 Stability and Coherence over Time

Interval evaluation through 45 seconds. We extend generation to 180 latent frames (approximately 45 seconds, or $8.6\times$ the 21-frame training rollout). Table 8 compares AdaState with α=4 against Self-Forcing on a 30-prompt subset of MovieGenBench Extended, using disjoint temporal intervals of the same rollouts.

AdaState shows greater robustness to temporal degradation, sustaining substantially more motion and a higher average score than Self-Forcing through the final interval. Its advantage in aesthetic quality

Table 6: Representative prompts from the 5-second evaluation set.

Prompt
A stylish woman strolls down a bustling Tokyo street, the warm glow of neon lights and animated city signs casting vibrant reflections. She wears a sleek black leather jacket paired with a flowing red dress and black boots...
A 3D animation of a small, round, fluffy creature with big, expressive eyes exploring a vibrant, enchanted forest. The creature, a whimsical blend of a rabbit and a squirrel, has soft blue fur and a bushy, striped tail...
A winter scene in a snowy forest, where a litter of playful golden retriever puppies emerge from the snow. Their heads pop out, their fluffy fur glistening in the sunlight, and they wag their tails joyfully...
A dramatic and dynamic moment captured in a realistic photographic style, featuring a golden retriever dog leaping into a pool to rescue a child. The dog is mid-jump, its legs stretched forward and its fur glistening in the sunlight...
A scenic photograph capturing the moment a steam train departs from the Glenfinnan Viaduct, a historic railway bridge in Scotland. The train moves gracefully over the arch-covered viaduct, its smoke billowing into the air...

Table 7: Representative prompts from the 30-second evaluation set.

Prompt
A sweeping aerial drone shot above dramatic coastal cliffs at golden hour, deep blue waves crashing against the rocks below and sending up white spray. The camera glides along the coastline at altitude, revealing rugged stone faces...
A dynamic FPV aerial flight through a sprawling cyberpunk megacity at night, neon billboards and holographic advertisements glowing pink, blue, and violet. The camera weaves between towering skyscrapers...
A dynamic FPV aerial flight over the African savanna at golden hour, the camera flying low and fast above tall yellow grasses dotted with acacia trees. Herds of elephants and giraffes move slowly across the plains...
A dynamic FPV aerial flight through a vibrant underwater coral city, where colorful corals line the streets and ancient stone ruins rise from the seabed. The camera moves swiftly between schools of tropical fish...
A sweeping aerial drone shot above the Scottish Highlands, the camera gliding over rolling green moors, mirror-still lochs, and the ruins of an ancient stone castle perched on a small island in the water...

also persists as the rollout progresses, although Self-Forcing retains higher subject and background consistency and motion smoothness. The adaptive reference thus supports continued scene evolution over extended rollouts, while remaining susceptible to gradual losses in visual detail and consistency.

Failure cases and the 45-second results. The dominant observed failure is gradual loss of local detail, including deformation of small non-rigid subjects such as fish around occlusion or re-entry. Table 8 quantifies the associated aggregate degradation: AdaState's imaging quality falls from 0.723 in the first interval to 0.630 in the last, and background consistency falls from 0.964 to 0.940, even while dynamic degree remains 0.796. These results suggest that sustaining scene progression and preserving fine visual detail remain distinct challenges over longer rollouts.

Distributional stability over time. We additionally evaluate disjoint five-second windows within 30-second rollouts on 30 prompts per method. W1, W3, and W6 denote 0–5, 10–15, and 25–30 seconds of the same rollout; W1 is not the separate 5-second benchmark. Using matched Wan2.1-T2V-1.3B samples as the reference distribution, we compute a PCA-projected FVD diagnostic. To reduce the feature dimension for small-sample covariance estimation [19], I3D features are projected onto an eight-dimensional PCA basis fit only on the reference set. Each prompt contributes multiple clips to the generated distribution. We use this measure to compare how the generated video distribution evolves across temporal windows relative to the same reference.

Table 8: **Long-horizon stability through 45 seconds.** Interval-level VBench scores on 30 prompts; all metrics are higher-is-better. AdaState (α=4) is labeled in bold; bold values indicate the better method for each metric and interval.

		Generation interval			
Metric	**Method**	**0–5 s**	**5–15 s**	**15–30 s**	**30–45 s**
Subject consistency	Self-Forcing	0.974	**0.964**	**0.968**	**0.973**
	AdaState	**0.975**	0.954	0.946	0.951
Background consistency	Self-Forcing	**0.967**	**0.958**	**0.958**	**0.966**
	AdaState	0.964	0.949	0.939	0.940
Motion smoothness	Self-Forcing	**0.988**	**0.989**	**0.990**	**0.989**
	AdaState	0.986	0.982	0.980	0.980
Dynamic degree	Self-Forcing	0.689	0.440	0.338	0.242
	AdaState	**0.811**	**0.933**	**0.850**	**0.796**
Aesthetic quality	Self-Forcing	0.641	0.589	0.540	0.499
	AdaState	**0.647**	**0.606**	**0.548**	**0.525**
Imaging quality	Self-Forcing	0.699	0.683	**0.648**	**0.643**
	AdaState	**0.723**	**0.698**	0.645	0.630
Six-axis average	Self-Forcing	0.826	0.770	0.740	0.719
	AdaState	**0.851**	**0.854**	**0.818**	**0.804**

Table 9: Generator-referenced PCA-FVD (↓) on 30 prompts per method. Δ is the rounded relative change from W1 to W6.

Method	Dyn.↑	W1	W3	W6	Δ
Self-Forcing	0.392	177.4	331.8	479.0	$+170\%$
Causal Forcing	0.933	245.7	408.8	627.7	$+155\%$
Infinity-RoPE	0.267	227.7	265.7	238.0	$+5\%$
MemRoPE	0.718	338.3	370.4	326.1	-4%
AdaState	0.922	242.5	242.4	217.6	-10%

AdaState has the lowest final-window estimate among these methods. The estimates rise markedly for Self-Forcing and Causal Forcing, whereas the other persistent-reference methods change less. These results complement the temporal and perceptual evaluations, showing that AdaState maintains a stable distributional relationship to the reference as generation progresses.

Motion-compensated coherence. Table 10 reports optical-flow magnitude and flow-normalized warp residual in the first and final five-second windows. Optical-flow magnitude measures the estimated displacement of image content between consecutive frames, with larger values indicating greater motion. The warp residual measures the discrepancy that remains after aligning consecutive frames using the estimated optical flow, following the use of warping error to assess temporal consistency [15]. Normalizing this discrepancy by flow magnitude expresses the remaining error relative to the amount of motion; lower values indicate better agreement between frame changes and the estimated motion.

AdaState's flow rises from 2.22 to 5.10 while its normalized residual falls from 1.41 to 0.85, indicating that increased motion is accompanied by stronger motion-compensated agreement. Causal Forcing has a lower final normalized residual alongside more than twice AdaState's flow magnitude, highlighting the importance of reading the two measures together. Combined with VBench smoothness, flickering, consistency, and the user study, these results support interpreting AdaState's dynamics gain as sustained, temporally coherent motion.

Table 10: Optical-flow magnitude and flow-normalized warp residual on the 30-prompt interval evaluation. Residual is lower-is-better; flow magnitude measures motion and is not itself a quality ranking.

Method	Flow W1	Flow W6	Residual W1↓	Residual W6↓
Infinity-RoPE	0.93	0.54	2.43	3.33
Self-Forcing	2.76	0.78	1.04	2.67
MemRoPE	2.33	3.03	1.15	0.86
AdaState	2.22	5.10	1.41	0.85
Causal Forcing	3.70	10.50	0.94	0.68

D.2 Cached-State Interventions

To test whether the cached state actively guides subsequent generation, we compare two 40-chunk rollouts for each prompt and random seed: one proceeds uninterrupted, while the other receives a single cached-state intervention at chunk n=20. We either zero the cached state, replace it with a cached state randomly selected from another chunk of the same video, or replace it with norm-matched noise. Generation then continues with the usual state rewrite at every subsequent chunk. Table 11 reports DINOv2 [22] feature similarity between the intervened and uninterrupted outputs at chunk $n+k$. Lower similarity indicates a greater change in the generated scene relative to the uninterrupted rollout, rather than necessarily lower video quality.

Table 11: DINOv2 [22] feature similarity to an uninterrupted rollout after a one-time intervention at chunk 20.

Intervention	k=1	k=2	k=4	k=8
Zeroed state	0.988	0.969	0.938	0.899
Swapped state	0.991	0.974	0.939	0.882
Norm-matched noise	0.969	0.943	0.907	0.874

All three interventions produce growing divergence from the uninterrupted rollout, showing that the cached state influences how the scene subsequently evolves. Norm-matched noise preserves the state's magnitude while replacing its contents, indicating that the stored information matters beyond the presence or scale of the state slot. The generated trajectory remains different even after subsequent state updates, while VBench subject consistency changes by at most 0.003 relative to the uninterrupted rollout. Together, these results support the adaptive state's role as a functional scene reference whose influence extends beyond the immediately following chunk. This influence can propagate through both subsequent states and generated content carried in the cache.

E Compute, Decoding, and Initialization

Inference protocol. We measure inference on one NVIDIA L40S at 5 seconds (21 latent frames, 81 output frames, 7 autoregressive chunks) and 30 seconds (120 latent frames, 477 output frames, 40 chunks). Resolution, batch size, and remaining inference settings follow the Self-Forcing setup [12]. Self-Forcing uses its original 21-frame KV window; AdaState uses an eight-frame attention-visible window with one live state frame.

Table 12: Inference cost on one L40S. Total time includes VAE decoding; denoising excludes it. FPS counts decoded output frames per total second.

Horizon	Method	Total/clip (s)↓	Denoising (s)↓	Peak memory (GB)↓	FPS↑
5 s	Self-Forcing	12.08±0.06	7.01	23.68	6.71
5 s	AdaState	13.95±0.04	8.88	20.20	5.81
30 s	Self-Forcing	81.33±0.10	51.53	27.77	5.87
30 s	AdaState	84.12±0.04	54.39	24.30	5.67

Relative to Self-Forcing, AdaState reduces peak memory by 14.7% at 5 seconds and 12.5% at 30 seconds. Denoising overhead falls from 26.7% to 5.6%, and total overhead from 15.5% to 3.4%. These savings reflect the tighter context window, which more than offsets the extra live state frame in memory. AdaState first denoises the adaptive state at $n = 1$ and reaches its full eight-block attention window one chunk later, at $n = 2$ (the third chunk). With their windows full, AdaState takes 1379.7$\pm$0.4 ms per chunk and Self-Forcing takes 1348.2$\pm$0.2 ms, a 2.3% gap. The larger early overhead amortizes over longer rollouts.

VAE decoding. We accumulate the generated content latent sequence and invoke Wan-VAE once after the autoregressive rollout; content chunks are not decoded immediately as they are generated. Internally, the temporally causal VAE processes the sequence in sequential chunks and propagates causal-convolution feature caches between them, rather than decoding all latent frames in one parallel operation. State latents are excluded, so the state incurs no additional VAE-decoding cost. Across 128 videos, separately measured VAE-only latency is 5.01 s for 81 output frames and 29.67 s for 477 output frames (61.9 and 62.2 ms per frame).

Initialization and marginal adaptation cost. AdaState initializes from the Self-Forcing DMD checkpoint and trains for 1000 iterations in 10.75 hours on two H200 GPUs (21.5 H200 GPU-hours). We also initialize directly from the earlier ODE-distilled causal checkpoint used before Self-Forcing's DMD stage and train for 1000 iterations in 10.75 hours on eight H200 GPUs (86 H200 GPU-hours). These runs match iteration count, not GPU-hour cost. All configurations share prior ODE distillation. Self-Forcing reports approximately 1.5 hours on 64 H100 GPUs for its DMD stage [12], or approximately 96 H100 GPU-hours; this prior cost is separate from AdaState adaptation. H100 and H200 measurements are not summed or treated as directly interchangeable.

Table 13: Initialization comparison at 5 seconds. Both AdaState variants train for 1000 iterations. Adaptation costs exclude prior ODE distillation and any prior Self-Forcing DMD.

Configuration	Adaptation (H200 GPU-h)	Subj.↑	Dyn.↑	Aesth.↑	Img.↑
Self-Forcing baseline	–	0.966	0.635	0.619	0.697
AdaState from Self-Forcing	21.5	0.961	0.828	0.630	0.702
AdaState from ODE-distilled	86	0.948	0.794	0.622	0.691

Direct ODE-distilled initialization improves dynamics over the original Self-Forcing baseline, while initialization from Self-Forcing produces stronger results across all four evaluated metrics. AdaState therefore benefits from Self-Forcing initialization without requiring it. Self-Forcing's DMD stage uses a larger training batch size, whereas AdaState adapts the pretrained model with a smaller batch and 21.5 H200 GPU-hours of additional training.

F Supplementary Video Results

We provide supplementary videos as part of this submission and on the project webpage at `adastate.github.io`, complementing the figures and analyses in the paper.

5-second generation. AdaState generations within the training horizon (21 frames), spanning diverse prompts including close-up nature shots, character animation, fantasy scenes, and stylized content. These demonstrate that AdaState produces coherent scene progression with natural camera motion and subject action at the within-horizon regime.

12-second generation. AdaState generations beyond the training horizon (51 frames), including cinematic portraits, aerial views, and detailed close-ups. These demonstrate that AdaState maintains temporal coherence and scene progression well past the 5-second training horizon without drift or freezing.

Ablation: state size and window. Side-by-side comparisons on two prompts showing three configurations: F_s=3 with W_p+F=9 (more state slots, wider window), F_s=1 with W_p+F=9 (single slot, wider window), and F_s=1 with W_p+F=6 (single slot, tight window, our final configuration).

The videos confirm the proportional design rule from Table 2: concentrating the state at a single recurrent position in a tight window produces the strongest dynamics.

Ablation: horizon weight α. Two panels contrasting short-horizon and long-horizon behavior. At 5 seconds, α=2 produces the richest dynamics (matching its peak in Table 2). At 30 seconds, α=4 yields the most stable long-horizon quality, confirming the two-regime training strategy described in Section 4.3.

30-second comparisons. Side-by-side comparison panels at 6× the training horizon (120 frames), each showing four methods spanning the anchor categories: no anchor, static anchor, EMA anchor, and AdaState. The prompts are aerial drone shots over coastal cliffs, an underwater coral city, and a cyberpunk megacity at night. These comparisons make the consistency-dynamics tradeoff from Figure 5 visually concrete: baselines either freeze the scene (static/EMA anchors) or accumulate artifacts (no anchor), while AdaState alone maintains both dynamics and coherence across the full 30-second rollout.

G Attention Analysis Details

This section provides the full methodology for the attention mass analysis summarized in Section 3.1 and visualized in Figure 2(a) of the main paper.

Notation. Let $\mathcal{A}_{q,k}^{\ell,h}$ denote the post-softmax attention weight from query token q to key token k at layer ℓ and head h. Let L_{frm} denote the number of tokens per frame (1560 in our setting). At generation chunk c, the visible K-window covers F_c frames; the f-th K-frame spans tokens $k \in [f \cdot L_{\text{frm}},\ (f{+}1) \cdot L_{\text{frm}})$.

Per-record frame mass. Each record corresponds to a unique (ℓ, c, t, p) tuple of layer, chunk, denoising step, and prompt. For a given record, the attention mass on K-frame f, averaged over heads H and a uniformly-spaced query subsample $\mathcal{Q}$ of size $|\mathcal{Q}|$=32, is:

$$\bar{m}_f^{\ell,c,t,p} = \frac{1}{|\mathcal{Q}| \cdot H} \sum_{q \in \mathcal{Q}} \sum_{h=1}^{H} \sum_{k=f \cdot L_{\text{frm}}}^{(f+1) \cdot L_{\text{frm}}-1} \mathcal{A}_{q,k}^{\ell,h} \tag{6}$$

By construction, $\sum_{f=0}^{F_c-1} \bar{m}_f^{\ell,c,t,p} = 1$.

Off-diagonal partition. The current chunk occupies the last n_{self}=3 K-frames. The off-diagonal (cached past) covers the first $n_{\text{off}}^{(c)} = F_c - n_{\text{self}}$ K-frames, i.e. the cache depth. We define the off-diagonal mass as $M_{\text{off}}^{\ell,c,t,p} = \sum_{f=0}^{n_{\text{off}}^{(c)}-1} \bar{m}_f^{\ell,c,t,p}$ and compute the renormalized off-diagonal share:

$$s_f^{\ell,c,t,p} = \frac{\bar{m}_f^{\ell,c,t,p}}{M_{\text{off}}^{\ell,c,t,p}}, \qquad \sum_{f=0}^{n_{\text{off}}^{(c)}-1} s_f^{\ell,c,t,p} = 1 \tag{7}$$

The renormalization treats the entire current chunk as self-attention (rather than excluding only the diagonal element), placing all cached positions on equal footing regardless of how much total mass goes to self-attention.

Aggregation. For cache depth $n^* \in \{3, 6, 9, 12, 15, 18\}$, we aggregate over the record set $\mathcal{R}_{n^*} = \{(\ell, c, t, p) : n_{\text{off}}^{(c)} = n^*\}$:

$$\bar{s}_{n^*,f} = \frac{1}{|\mathcal{R}_{n^*}|} \sum_{(\ell,c,t,p) \in \mathcal{R}_{n^*}} s_f^{\ell,c,t,p}, \qquad f \in [0, n^*) \tag{8}$$

Records at t=0 (the cache rerun with clean inputs) are excluded, as they reflect a different attention regime from sampling-time noisy queries.

Analysis Setting. We probe the Self-Forcing DMD checkpoint with $P{=}5$ prompts, 21-frame generation (7 chunks), chunk size $n_{\text{self}}{=}3$, $K{=}4$ denoising steps, $L{=}30$ layers, $H{=}12$ heads averaged, and query subsample $|\mathcal{Q}|{=}32$. Total records with cache-rerun excluded: 3,600 (600 per cache depth). The anchor rank is consistent across layers (mean rank 2–3, std < 1), confirming the pattern is not driven by a subset of layers.

H Limitations and Future Work

We briefly discuss the limitations in the main paper. The extended-rollout analysis in Appendix D.1 shows that AdaState sustains scene progression through 45 seconds, while preserving fine visual detail remains a challenge as generation continues. Small, non-rigid subjects can gradually deform, particularly around occlusion or re-entry, motivating further exploration of how recurrent states retain and update scene information. Below, we discuss state capacity and broader applicability as directions for future work.

State capacity. Our ablations (Table 2) show that $F_s{=}1$ is sufficient at the current training horizon of 21 frames. However, as rollout length grows, a single frame may not carry enough information to serve as a complete scene reference, particularly for scenes with multiple actors, concurrent events, or parallel narrative threads. Future work could explore multi-slot states with structured roles, or augment the adaptive state with explicit external memory that the model can read and write through attention.

Broader applicability. The attention-sink pattern that motivates AdaState has been characterized broadly in causal transformers [29], suggesting the adaptive state formulation could apply to other autoregressive generation domains (e.g. audio, 3D, or long-context language generation) wherever a fixed anchor position dominates attention. We leave cross-domain investigation as future work.

I Broader Impact and Ethics

Broader impact. AdaState advances streaming video generation by enabling richer temporal dynamics, camera motion, scene progression, and subject action, within a lightweight fine-tuning framework. This has direct applications in creative tools, real-time content generation, and interactive media, where coherent long-horizon video is currently limited by the consistency-dynamics tradeoff. The method requires only 1000 fine-tuning iterations on two GPUs, making it accessible to academic and small-scale research groups.

Potential risks. As with any improvement in generative video quality, increased realism and temporal coherence could lower the barrier for producing disinformation or non-consensual deepfake content. We emphasize that AdaState does not introduce new generation capabilities, it improves the dynamics of an existing open-source model (Wan 2.1-1.3B), and the same safeguards applicable to the base model (content filtering, watermarking, usage policies) remain applicable. We encourage the community to develop and deploy provenance-tracking and watermarking tools alongside advances in generation quality.

Data and human subjects. Our training uses only text prompts from publicly available benchmarks (MovieGenBench [24]) and does not involve any personal, sensitive, or copyrighted visual data. The user study was conducted on the Prolific platform with informed consent; participants were compensated at a rate above the platform's recommended minimum. No personally identifiable information was collected beyond anonymous Prolific IDs.

LLM usage. We used large language models to assist with standard engineering tasks such as codebase setup, experiment scripting, and evaluation pipelines. LLMs were not used in the design of the core method, the formulation of the adaptive state mechanism, or the analysis of experimental results.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract clearly states the area of the problem, the identified gap and the proposed solution.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are provided in two places. One in the discussion section of the main paper and in the appendix in a more detailed form.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: While we do not include any theoretical results, we provide the overall algorithm proposed, ablations on the hyperparameters and the methodology for our analysis. Analysis details on attention distribution and algorithm definition is provided in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide our evaluation prompts, and details of our evaluation protocol as a part of our supplementary material. In addition, we provide the details of training (hyperparameters and setup) in detail in appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We provide the training algorithm description in the supplementary material. The code will be made publicly available upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide ablations on each of the hyperparameter chosen in the method. Additionally, we provide training hyperparameters such as learning rate and effective batch size in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: While the error bars are not reported in the main paper due to page limitations, we provide a detailed breakdown of our benchmark scores in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide our hardware specifications in the experiments section and in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We carefully reviewed and conform with NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We include an impact statement as a part of our supplementary material.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We train AdaState with prompts that are publicly available. Thus our results do not imply an additional risk compared to the available dataset and the baseline model we fine-tune.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: This paper only utilizes generated assets, where the generation prompts are constructed from publicly available datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We provide the details of our user study in detail as a part of our supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We conduct a user study with 40 anonymous participants, using the Prolific platform.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We include a related statement as a part of our supplementary material.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.