REVISE BY APPENDING: CORRECTING COMMITTED CHUNKS IN AUTOREGRESSIVE VIDEO DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Chunk-wise autoregressive video diffusion models generate a video a few latent frames at a time and keep the key–value (KV) states of every finished chunk for the chunks that follow. Each chunk is therefore final once it is written. When a later chunk shows that an earlier one set up an event wrongly, the model can only continue from that mistake, and prompts that describe several consecutive events often end incomplete. Overwriting the earlier latent would invalidate every KV state computed from it. We observe that only the latent the Transformer attends to has to stay fixed, while the frames shown to the viewer can be decoded from the whole generated record. AmendAR uses this separation to correct earlier chunks by appending. After each video chunk, the same Transformer samples a small revision latent that describes how that exact chunk should change. The revision's KV states are appended to the cache, so later chunks are generated in agreement with the correction, and a lightweight decoder scans the record from the last chunk to the first, adding a residual correction to each earlier latent before VAE decoding. Revisions are learned from enhanced versions of the model's own drafts. We prove that finalizing a chunk early discards exactly the information that later chunks carry about it, and that the size of the decoder state bounds how many correction directions reach earlier chunks. On Wan2.1, MAGI-1, and HunyuanVideo 1.5 backbones, AmendAR improves event completion on VideoPhy-2, TC-Bench, and StoryEval by 4.8–8.1 points, including StoryEval from 24.9% to 33.0% on Wan, with 9–21% additional latency.

1 INTRODUCTION

Chunk-wise autoregressive video diffusion models, such as Self Forcing on Wan2.1, MAGI-1, and minWM on HunyuanVideo 1.5 (Huang et al., 2025; Sand.ai et al., 2025; Zhao et al., 2026), generate a video a few latent frames at a time. Each chunk is denoised while attending to the key–value (KV) cache of retained earlier chunks, and its own KV states are then appended to the cache. This makes streaming generation fast, and it also makes every chunk final as soon as it is written. The cost appears on prompts that describe several consecutive events (Figure 1). Whether "a man picks an apple and places it in a basket" succeeds depends on early chunks, which fix where the hand and the basket are, and these chunks are generated before the chunks in which the event has to finish. When a later chunk reveals that this setup cannot reach the requested outcome, the model can only continue from it. In our evaluation, Self Forcing completes 24.9% of events on StoryEval (Wang et al., 2025b) (Table 1).

Current methods improve how the next chunk is generated. Causal distillation and training on the model's own rollouts reduce the mismatch between training and inference (Yin et al., 2025; Huang et al., 2025; Cui et al., 2026; Zhu et al., 2026), and rolling-window sampling keeps a few recent chunks editable until they leave the window (Liu et al., 2026). After a chunk is released, later chunks change only their own frames. The direct remedy, overwriting the earlier latent, breaks the cache: every later KV state was computed from the old latent, so exact reuse requires recomputing those states, and the later chunks themselves were generated for the old content. Enhancing a generated video with a separate model (He et al., 2024) can repair frames, but these post-processing corrections do not update the generator's cached history for future chunks.

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



Figure 1: Prompts with consecutive events. Ordered frames for six prompts: a basketball throw, a hat taken by an eagle, an apple placed in a basket, a ball passed between hands, melting ice cream, and a thawing forest. Each prompt succeeds only if the early frames set up a state from which the later event can finish, which is the dependency a chunk-wise generator cannot revisit once the early chunks are written.

We start from a simple observation. A chunk-wise generator uses each latent for two purposes: later chunks attend to it, and the VAE decodes it into the frames the viewer sees. Only the first purpose requires the latent to stay fixed. If the displayed frames of chunk i are decoded from the whole record, information appended after chunk i can change how chunk i looks while the cache stays exactly as it was. Correction then becomes an append operation.

AmendAR (**amend**ing **a**uto**r**egressive outputs) implements this idea with three components (Figure 2). After each video chunk x^i, the same Transformer samples a small revision latent r^i that describes how this finished chunk should change, and appends its KV states. Later video chunks attend to earlier revisions, so they are generated in agreement with corrections already decided. At rendering time, a lightweight decoder scans the record from the last chunk to the first with a fixed-size state and adds a residual correction to each earlier latent before VAE decoding. Training targets come from enhancing the exact draft the current model produced, so each revision learns what to change in that particular draft.

Our analysis makes these choices precise. Finalizing chunk i at step s loses exactly the squared-error reduction that the latents generated after step s provide about chunk i (Theorem 1). Appending leaves every stored KV state unchanged, and a decoder state of m scalars passes at most m independent correction directions across a chunk boundary (Proposition 2). On Wan, MAGI, and Hunyuan backbones from 1.3B to 8.3B parameters, AmendAR improves VideoPhy-2, TC-Bench, and StoryEval completion by 4.8–8.1 points at 9–21% additional latency and 4.4–8.3% additional KV memory. Most of the gain appears when the decoder re-renders earlier chunks, AmendAR exceeds video-only training with matched learning time in all nine backbone–benchmark pairs, and a same-draft comparison places the remaining gap to the enhancement teacher in sampling the revisions.

2 Related Work

Training chunk-wise video generators. CausVid distills a bidirectional teacher into a causal few-step generator (Yin et al., 2025), and MAGI-1 trains chunk-wise autoregression at scale (Sand.ai et al., 2025). Self Forcing trains the student on its own sampled history (Huang et al., 2025), Self-Forcing++ extends this to minute-long rollouts supervised on shorter segments (Cui et al., 2026), and Causal Forcing improves the teacher and initialization for causal distillation (Zhu et al., 2026). These methods make each new chunk more reliable; AmendAR adds a way for later chunks to change the frames of chunks already written.

Managing the history of long rollouts. Rolling Forcing denoises several chunks jointly inside a moving window (Liu et al., 2026). LongLive combines long-rollout training with persistent and recent context (Yang et al., 2026), Context Forcing trains with a long-context teacher and selects memory (Chen et al., 2026), and Deep Forcing changes sink retention and cache compression without further training (Yi et al., 2026). Pathwise test-time correction adjusts the sampling trajectory of the current chunk (Xiang et al., 2026). These methods decide what later chunks can see and how long a chunk stays editable. AmendAR adds a path from any later chunk to the rendered output of any earlier chunk, including chunks far outside a window.

Refining generated visual content. TensorAR adds refinement steps to autoregressive image generation (Cheng et al., 2026), and visual self-refinement revisits autoregressive visual predictions (Wang et al., 2025a). VEnhancer maps a generated video clip to an enhanced one (He et al., 2024). AmendAR performs refinement inside causal video generation: the refinement is itself sampled by the generator, stored in the cache so that later chunks condition on it, and applied to earlier chunks by a backward decoder. We use VEnhancer offline to build training targets.

3 Why Chunks Become Final and What Later Chunks Could Fix

Chunk-wise generation. Given a text prompt c, a chunk-wise generator produces latent chunks $x^1, \dots, x^N$, each spanning a few latent frames (three on Wan). Chunk x^i is sampled by a few flow-matching steps whose queries attend to the cached keys and values of $x^{<i}$; a clean forward pass then appends the KV states of x^i to the cache. The VAE decodes each x^i into frames. The frames of chunk i are therefore fixed at step i, and all later computation affects only later frames.

Goal. After k chunks have been generated, we want rendered latents $\hat{x}^{i|k}$ for every $i \le k$ that may use everything generated up to chunk k. Two constraints keep the generator practical: stored latents and their KV states are never modified, so the cache remains valid for reuse, and the per-chunk work of the generator stays close to that of the native model. Overwriting x^i violates the first constraint, because every later KV state that attended to x^i becomes stale and the later chunks were sampled for the old content (Appendix C).

What finalizing early costs. Let y^i be a target for chunk i (in our training, an enhanced version of the generated chunk), and let $\mathcal{G}_k = \sigma(c, x^{\le k}, r^{\le k})$ be the information in the record after k steps, where r^k is the revision latent introduced in Section 4 (empty for a native model). The best squared-error estimate of y^i from this record is $\mu^i_k = \mathbb{E}[y^i \mid \mathcal{G}_k]$.

Theorem 1 (Cost of finalizing early)**.** *Let $\hat{y}^i_{\mathrm{c}}$ be any estimate of y^i computed from $\mathcal{G}_s$ (finalized at step $s \ge i$) and $\hat{y}^i_{\mathrm{a}}$ any estimate computed from $\mathcal{G}_N$ (amended at the end), with squared-error risks $\mathcal{R}^i_{\mathrm{c}}$ and $\mathcal{R}^i_{\mathrm{a}}$. Then*

$$\mathcal{R}^i_{\mathrm{c}} - \mathcal{R}^i_{\mathrm{a}} = \sum_{k=s+1}^{N} \left(V^i_{k,x} + V^i_{k,r}\right) + \mathcal{E}^i_{\mathrm{c}} - \mathcal{E}^i_{\mathrm{a}}, \tag{1}$$

where $V^i_{k,x}$ and $V^i_{k,r}$ are the expected squared changes of the best estimate of y^i when x^k and then r^k are revealed, and $\mathcal{E}^i_{\mathrm{c}} = \mathbb{E}\|\hat{y}^i_{\mathrm{c}} - \mu^i_s\|^2$, $\mathcal{E}^i_{\mathrm{a}} = \mathbb{E}\|\hat{y}^i_{\mathrm{a}} - \mu^i_N\|^2$ are the estimators' fitting errors.

The sum is nonnegative and requires no independence between chunks. It measures exactly how much later chunks move the best estimate of chunk i, and finalizing at step s discards all of it. Appendix A gives the proof and splits $\mathcal{E}^i_{\mathrm{a}}$ into information the decoder cannot see and error in fitting the information it sees. Eq. (1) leads to three design decisions. The record should contain content generated specifically to inform corrections, which contributes the terms $V^i_{k,r}$; this is the revision latent. An estimator must read the later record to collect the sum; this is the backward decoder. The fitting error $\mathcal{E}^i_{\mathrm{a}}$ decides how much of the sum is realized, so the decoder must be trained against targets that are well defined for each draft.

4 AmendAR: Correcting Earlier Chunks by Appending

AmendAR interleaves each video chunk x^i with a revision chunk r^i in the generated record and keeps the cache append-only (Figure 2). The generator samples x^i and then r^i (Section 4.1); a small

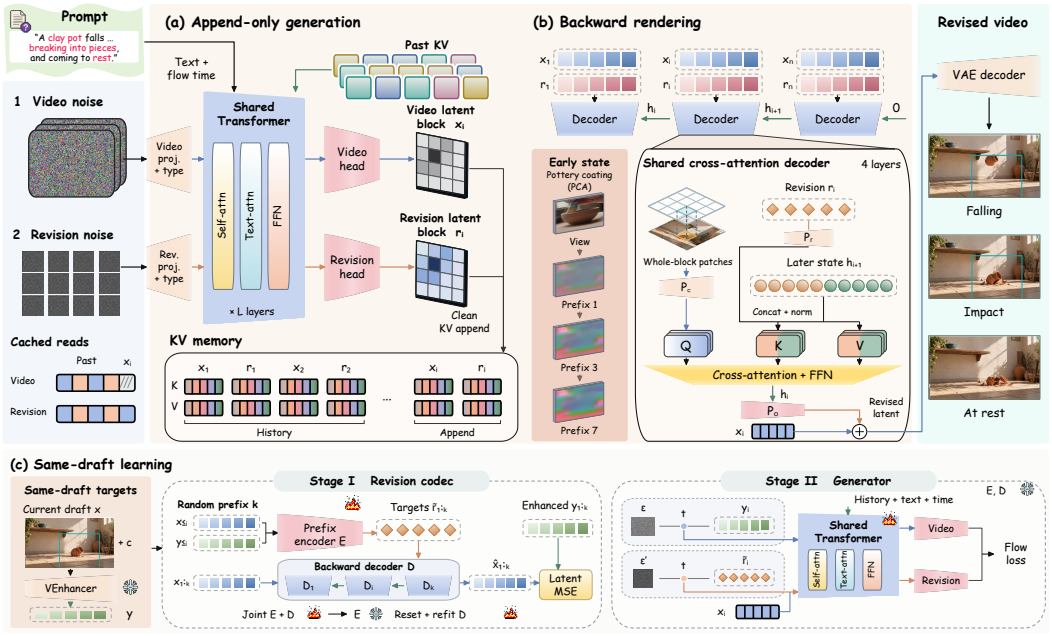


Figure 2: AmendAR overview. (a) A shared Transformer samples each video chunk x^i and then its revision r^i with separate projections, and appends the clean KV states of both without rewriting the stored history. (b) A shared cross-attention decoder reads the record from the last chunk to the first; the local revision features u^i and the incoming state h^{i+1} produce a residual added to x^i before VAE decoding. The inset shows how an early state changes as the prefix grows. (c) Enhancing the model's own draft provides aligned targets for the prefix encoder, codec fitting, decoder refitting, and flow learning of both branches.

decoder reads the record backward and renders corrected video latents (Section 4.2); and both are trained on targets aligned with the model's own drafts (Section 4.3). Algorithm 1 lists every step.

4.1 Sampling a revision after each finished chunk

A revision has to describe the draft that was actually produced, so r^i is sampled after x^i is complete. With the record $\mathcal{H}^{<i} = (x^1, r^1, \ldots, x^{i-1}, r^{i-1})$, the generator factorizes as

$$p_\theta\big(x^{1:N}, r^{1:N} \mid c\big) = \prod_{i=1}^{N} p_\theta\big(x^i \mid \mathcal{H}^{<i}, c\big)\, p_\theta\big(r^i \mid \mathcal{H}^{<i}, x^i, c\big). \tag{2}$$

Each factor is a separate flow-matching sampler with its own noise. Denoising x^i and r^i jointly would form the revision while the video chunk is still noisy, so the revision could not refer to the exact content that the viewer will see.

Append-only cache. Each step issues queries only for new rows. The video sampler's queries attend to the cached $\mathcal{H}^{<i}$; a clean pass appends the KV states of x^i; the revision sampler's queries attend to $\mathcal{H}^{<i}$ and x^i; a second clean pass appends the KV states of r^i. Every stored row keeps its input, position, and visible set, so its KV states stay exactly unchanged (Proposition 2) and no earlier query is recomputed. Both samplers share all Transformer blocks and the text and time conditioning. The only new generator parameters are two type embeddings and the input and output projections of the revision, which lives on a coarser grid: on Wan a revision has $n_r = 390$ tokens of 64 channels against $n_x = 4680$ video tokens, a ratio $\rho = n_r/n_x = 8.3\%$.

Forward path. Because $r^{<i}$ sits in the cache, video chunk i attends to all earlier revisions. After the decoder corrects chunk j, the viewer sees the corrected chunk; a later chunk that attended only to x^j would continue the uncorrected content, whereas attending to r^j lets it follow the correction. The value of this access has an exact form. For a future chunk z, its noisy version $z_t = tz + (1-t)\epsilon$,

the readable revision history $\mathcal{Q}$, and all other retained context $\mathcal{X}$, the optimal flow-matching loss drops by $\Delta(t) = \mathbb{E}\|\mathbb{E}[z \mid z_t, \mathcal{X}, \mathcal{Q}] - \mathbb{E}[z \mid z_t, \mathcal{X}]\|^2/(1-t)^2$, which is zero exactly when z is conditionally independent of $\mathcal{Q}$ given $\mathcal{X}$ (Appendix B). Section 5.3 measures this reduction for the trained generator.

4.2 Rendering earlier chunks from later information

The cache carries revisions forward; earlier frames need a path in the opposite direction. For a record of k chunks, the decoder projects each video chunk to queries $q^i = P_c(x^i)$ and each revision to features $u^i = P_r(r^i)$, and a shared cross-attention cell f_ϕ scans from the last chunk to the first:

$$\begin{aligned} h^{k+1} &= 0, & h^i &= f_\phi\big(q^i,\, [u^i; h^{i+1}]\big), \\ \hat{x}^{i|k} &= x^i + P_o\big(h^i\big), & i &= k, k-1, \ldots, 1. \end{aligned} \tag{3}$$

In f_ϕ, the queries q^i attend to the local revision features concatenated with the incoming state $[u^i; h^{i+1}]$, and P_o maps the state back to the video-latent layout before VAE decoding. The final video uses $\hat{x}^{i|N}$.

Each choice in Eq. (3) serves the goal in Section 3. The residual form keeps the draft's detail, so the revision encodes only the change and can stay small. The backward recurrence lets h^{i+1} summarize all later video and revision chunks, because every later cell reads its own x^j and r^j; a final rendering costs N evaluations of a four-layer, width-512 cell and no generator pass, and a streaming preview after chunk k costs k evaluations. The state $h^i \in \mathbb{R}^{n_h \times d_h}$ has a fixed size, so one decoder serves any length, and its size $m = n_h d_h$ caps how much correction can cross a chunk boundary:

Proposition 2 (Cache invariance and transmitted capacity)**.** *With fixed weights, positions, and visible sets for stored rows, appending new rows leaves all stored KV states unchanged. For any differentiable decoder of the form Eq. (3) and any boundary between chunks j and $j+1$, the Jacobian of the residual outputs $\{\hat{x}^{i|k} - x^i\}_{i \le j}$ with respect to the later inputs $\{x^l, r^l\}_{l > j}$ has rank at most m.*

The rank bound holds because every path from later inputs to earlier outputs passes through h^{j+1}; it is tight for time-varying linear scans (Appendix C). Section 5.4 varies m with the cell width fixed.

4.3 Learning revisions from enhanced versions of the model's own drafts

The decoder learns a correction to a specific draft, so its target must be the same video, improved. We roll out the current generator without gradients to obtain $(x^{1:N}, r^{1:N})$ and enhance exactly these frames with VEnhancer (He et al., 2024), giving $y^{1:N} = G^+(x^{1:N}, c)$. A different video that merely matches the prompt would place objects and motions elsewhere, and the residual $y^i - x^i$ would mostly encode those arbitrary differences. The teacher runs offline during training.

Stage I: revision codec. A causal encoder turns each draft prefix and its enhanced counterpart into a target revision $\tilde{r}^{\,i} = E_\psi(x^{\le i}, y^{\le i})$. The encoder and the decoder D_ϕ of Eq. (3) are fitted jointly on random prefix lengths,

$$\mathcal{L}_{\text{codec}}(\psi, \phi) = \mathbb{E}_{k \sim \mathcal{U}\{1,\ldots,N\}}\Big[\frac{1}{k}\sum_{i=1}^{k} \|\hat{x}^{i|k}(x^{1:k}, \tilde{r}^{\,1:k}) - y^i\|_{\text{av}}^2\Big], \tag{4}$$

where $\|\cdot\|_{\text{av}}^2$ averages over latent elements. Training on prefixes teaches one decoder to render with any amount of later context, as it must during streaming. We then freeze E_ψ, so each revision keeps a fixed meaning, and refit D_ϕ from its initialization.

Stage II: generator. With the codec fixed, both branches learn conditional flows (Lipman et al., 2023) with $t \sim \mathcal{U}[0,1]$ and independent $\epsilon, \epsilon' \sim \mathcal{N}(0, I)$:

$$\begin{aligned} \mathcal{L}_x(\theta) &= \mathbb{E}\,\|v_\theta^x\big(x_t^i, t, \mathcal{H}^{<i}, c\big) - (y^i - \epsilon)\|_{\text{av}}^2, & x_t^i &= t y^i + (1-t)\epsilon, \\ \mathcal{L}_r(\theta) &= \mathbb{E}\,\|v_\theta^r\big(r_t^i, t, \mathcal{H}^{<i}, x^i, c\big) - (\tilde{r}^{\,i} - \epsilon')\|_{\text{av}}^2, & r_t^i &= t\tilde{r}^{\,i} + (1-t)\epsilon'. \end{aligned} \tag{5}$$

We minimize $\mathcal{L}_x + \mathcal{L}_r$. The video branch learns the enhanced chunk while conditioning on its own generated history, and the revision branch learns to produce, from the record alone, the revision the encoder derived for the same draft x^i. Rollouts are regenerated as θ changes, so the targets describe

Table 1: Multi-benchmark native and external comparison. Means over three batches. VBench totals, CompBench scores $\times 100$, joint/transition/StoryEval completion are percentages; SA, PC, Q1 and Q2 retain their 1–5 scales. All task columns are higher-is-better. Time is final VideoPhy-2 latency (s); KV is persistent visual storage (decimal GB). Compare within backbone; external systems retain their training and history policies. †: independent Self-Forcing++ reproduction. Bold marks the best displayed mean per backbone (ties retained); red adds emphasis in the larger Wan group.

Method	VBench		CompBench		VideoPhy-2			TC-Bench			Story	Inference	
	Short	Long	Act.	Inter.	SA	PC	Joint	Q1	Q2	Succ.	Event	Time↓	KV↓
Wan2.1 / Self Forcing, 1.3B (Wan Team et al., 2025; Huang et al., 2025)													
Self Forcing (Huang et al., 2025)	81.95	78.34	54.9	61.2	3.43	3.23	25.5	2.67	3.06	16.8	24.9	11.55	6.04
Rolling Forcing (Liu et al., 2026)	81.86	80.53	56.3	61.7	3.47	3.24	27.2	2.72	3.09	18.1	27.6	13.26	**1.73**
Self-Forcing++† (Cui et al., 2026)	81.55	80.02	55.6	62.2	3.46	3.31	28.2	2.73	3.09	18.6	27.4	11.26	6.04
LongLive (Yang et al., 2026)	82.15	80.80	56.7	61.5	3.51	3.27	28.0	2.77	3.18	19.3	28.7	**10.58**	3.45
Causal Forcing (Zhu et al., 2026)	**82.40**	79.89	58.0	64.0	3.55	3.35	30.6	2.83	3.21	21.6	30.7	11.48	6.04
Context Forcing (Chen et al., 2026)	81.90	80.55	57.1	62.4	3.51	3.33	29.7	2.80	3.16	20.0	29.6	11.93	6.04
Deep Forcing (Yi et al., 2026)	81.98	80.74	55.3	61.2	3.44	3.26	26.3	2.68	3.11	16.8	25.3	11.94	6.04
Pathwise TTC (Xiang et al., 2026)	82.20	**81.23**	57.6	63.4	3.50	3.35	30.2	2.82	**3.23**	21.1	30.1	17.65	6.04
AmendAR (ours)	82.05	78.39	**59.7**	**65.5**	**3.56**	**3.45**	**33.2**	**2.91**	3.16	**24.1**	**33.0**	14.07	6.54
MAGI-1, 4.5B-distill (Sand.ai et al., 2025)													
Native A-only	**81.79**	78.37	52.4	58.2	3.38	3.26	24.7	2.69	3.02	17.3	23.9	**958.12**	**6.77**
AmendAR (ours)	81.78	**78.38**	**55.8**	**63.2**	**3.51**	**3.47**	**31.8**	**2.91**	**3.13**	**24.4**	**31.9**	1045.86	7.06
HunyuanVideo 1.5 / minWM, 8.3B (Wu et al., 2025; Zhao et al., 2026)													
Native A-only	78.97	75.67	47.3	53.9	3.11	3.05	17.7	2.46	2.90	12.0	18.6	**53.60**	**13.80**
AmendAR (ours)	**79.07**	**75.72**	**52.0**	**57.3**	**3.28**	**3.26**	**24.6**	**2.70**	**2.98**	**16.8**	**25.8**	61.81	14.66

the errors the current model makes. The encoder reads $y^{\leq i}$, which the teacher computed from the whole video, whereas the sampler sees only $\mathcal{H}^{<i}$ and x^i; Section 5.4 measures what this difference costs.

5 Experiments

We ask four questions. Does correcting earlier chunks raise event completion, and at what cost (Section 5.2)? Does the gain come from the revisions or from extra training toward enhanced targets, and do both the forward and the backward path contribute (Section 5.3)? What limits the correction (Section 5.4)?

5.1 Setup

Backbones. We adapt three chunk-wise generators: Wan2.1 with Self Forcing (1.3B, one A100-80GB) (Wan Team et al., 2025; Huang et al., 2025), MAGI-1 4.5B-distill (one A100) (Sand.ai et al., 2025), and HunyuanVideo 1.5 with minWM (8.3B, four A100s) (Wu et al., 2025; Zhao et al., 2026). Each is compared with its native model (Native A-only), which generates video chunks only, from the same initialization and output settings (Appendix D). On Wan we also compare seven published systems under their own training and history policies.

Benchmarks. Three endpoints measure whether requested events finish. VideoPhy-2 joint success requires both the semantic-adherence and the physical-commonsense score to reach four of five (Bansal et al., 2026); TC-Bench transition completion uses three human ratings per video (Feng et al., 2025); StoryEval scores the fraction of consecutive events completed (Wang et al., 2025b). T2V-CompBench V2 action and interaction binding (Sun et al., 2025), VBench and VBench-Long totals (Huang et al., 2024; 2026), and ChronoMagic (Yuan et al., 2024) check relations, visual quality, and motion. All benchmarks use their full sets (Appendix E). Internal comparisons use three paired training-and-generation seeds; tables report means, and the text reports paired 95% Student-t intervals where a difference is small.

5.2 Event completion and its cost

AmendAR raises completion on every backbone. In Table 1, AmendAR improves all three completion endpoints over each native model by 4.8–8.1 points: VideoPhy-2 joint success rises by 7.7, 7.1, and 6.9 points on Wan, MAGI, and Hunyuan, StoryEval by 8.1, 8.0, and 7.2, and TC-Bench

completion by 7.3, 7.1, and 4.8. Action and interaction binding improve as well (Wan: 54.9 → 59.7 and 61.2 → 65.5), while VBench totals change by at most 0.1 point. This pattern fits the mechanism: the correction changes how early chunks set up later events, which completion benchmarks score, and leaves per-frame quality, which VBench scores, at the backbone's level. Among Wan systems, long-rollout methods lead VBench-Long and Causal Forcing leads external completion; AmendAR exceeds it on StoryEval (33.0% vs. 30.7%; paired difference 2.4 points, 95% CI 2.0–2.7), TC-Bench (24.1% vs. 21.6%), and joint success (33.2% vs. 30.6%).

The cost is a small second sampler. Table 2 separates the preview, available once the first chunk is decoded, from the final output. The preview is almost unchanged (1.32 vs. 1.37 s on Wan) because the revision is sampled after its chunk. Final latency grows by 21%, 9%, and 18% on Wan, MAGI, and Hunyuan, and persistent KV memory grows by the token ratio ρ: 8.3%, 4.4%, and 6.3%. The latency overhead follows the same order as ρ, since revision sampling scales with the number of revision tokens and the backward scan adds only N small cell evaluations. In Figure 3, AmendAR on Wan reaches the highest success at 14.07 s, between Causal Forcing (11.48 s) and Pathwise TTC (17.65 s). Adaptation takes 522–3802 GPU-hours, mostly for building targets (Table 9).

Table 2: Native inference footprint. Three-batch means. Preview ends before revision; final includes complete rendering. Bold marks the lower mean within each backbone; ties are retained. Backbones and hardware are specified in Table 7.

Output	VBench-short (s)		KV (GB)
	Preview	Final	
Wan2.1 / Self Forcing, 1.3B			
Native A-only	**1.32**	**11.63**	**6.04**
AmendAR	1.37	14.06	6.54
MAGI-1, 4.5B-distill			
Native A-only	**188.05**	**952.56**	**6.77**
AmendAR	188.62	1039.09	7.06
HunyuanVideo 1.5 / minWM, 8.3B			
Native A-only	**8.60**	**52.78**	**13.80**
AmendAR	**8.60**	62.20	14.66

5.3 Where the gain comes from

Revisions add to what training on enhanced targets gives. AmendAR trains toward enhanced targets, which could improve a video-only generator by itself. Table 3 trains the native model on the same targets with an equal number of video updates, or with equal parameter-learning GPU time and target availability. Both controls improve over the native model (Wan StoryEval 24.9% → 30.7% under equal time), so aligned targets are useful on their own. AmendAR adds 2.2–3.8 points over the stronger, time-matched control in all nine backbone–benchmark pairs; the least certain margin, Wan VideoPhy-2, is 2.7 points (95% CI −0.3–5.6).

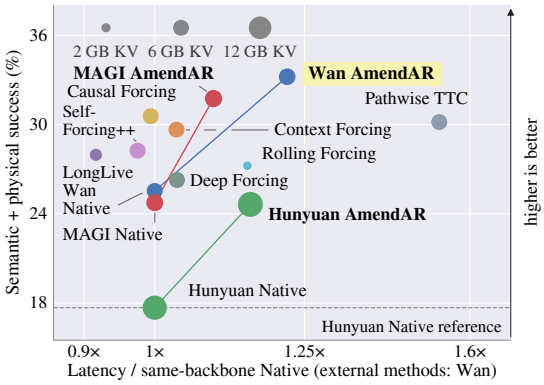


Figure 3: Event completion against inference cost. VideoPhy-2 joint success (both scores ≥ 4; 590 videos per seed, three seeds) versus final-output latency relative to the same-backbone native model. Bubble area is analytic persistent visual KV in decimal GB. Lines connect native/AmendAR pairs; external systems use Wan. Timing protocol: Appendix E.3.

Each component carries part of the correction. The lower group of Table 3 removes one component at a time. Refitting the decoder without the local revision u^i costs the most (3.2–4.6 points), although later video content still reaches each cell through the state; the revision therefore carries information that the video latents alone do not supply. Zeroing the incoming state h^{i+1}, so that each chunk is corrected by its own revision only, costs 1.9–3.9 points, which is the contribution of later chunks. Training the generator without video-to-revision attention costs 0.8–2.7 points, which is the contribution of the forward path to the drafts.

Both paths change task outcomes. Table 4 isolates each path. Forward: with video queries blocked from revision keys, raw drafts before decoding lose 1.9–3.3 points on every benchmark and backbone, so later chunks improve when they can read the revisions of earlier chunks. Figure 4 shows the three trajectories for one prompt.

Table 3: Matched learning and component controls. VideoPhy-2 (VP-2), TC-Bench (TC) and StoryEval completion percentages (↑), three-batch means. The first three rows compare the full method with extra A-only learning; the last three test components. Equal time matches parameter learning and teacher-target examples, not preparation. No-u refits the decoder; no-r-read retrains the generator; no incoming state sets $h^{i+1} = 0$ on fixed sequences. Model references and configurations are in Table 7.

Configuration	Wan2.1 / SF (1.3B)			MAGI-1 (4.5B-distill)			HY1.5 / minWM (8.3B)		
	VP-2	TC	Story	VP-2	TC	Story	VP-2	TC	Story
AmendAR	**33.2**	**24.1**	**33.0**	**31.8**	**24.4**	**31.9**	**24.6**	**16.8**	**25.8**
Equal A updates	28.3	20.2	28.9	28.0	20.5	27.6	20.6	14.8	21.8
Equal learning time	30.6	21.2	30.7	29.5	20.6	29.1	21.7	14.6	23.3
No local u	29.6	19.6	29.1	28.0	20.1	27.3	20.6	13.6	21.4
No video-to-r read	31.2	21.7	30.8	29.6	21.7	29.7	22.0	16.0	24.1
No incoming state	30.4	20.2	29.7	29.9	21.2	28.7	21.9	13.9	22.9

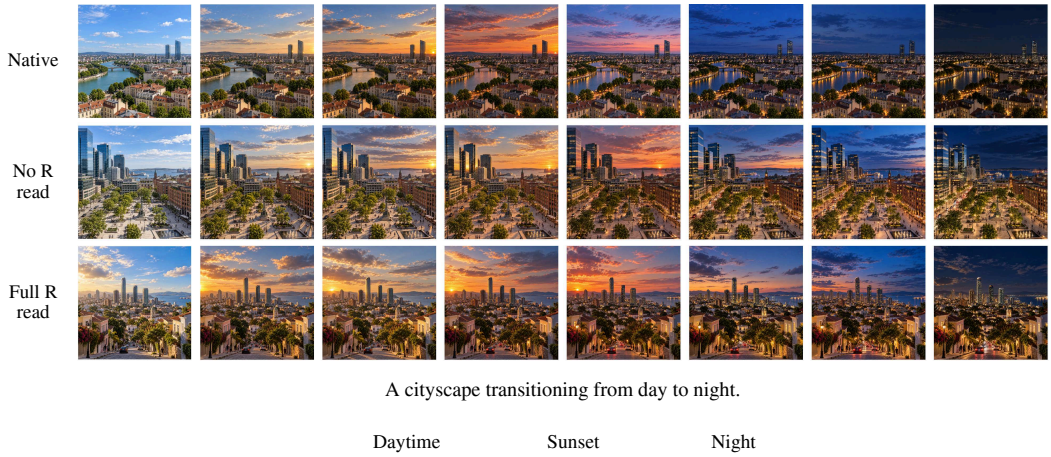


Figure 4: Forward conditioning on revisions. Prompt: "A cityscape transitioning from day to night." Rows show Native, the AmendAR generator with revision keys blocked (No R read), and with full access (Full R read), at eight ordered snapshots. The latter two rows are raw drafts before backward rendering. Each row is its own generated trajectory, so layouts differ; aggregate results are in Table 4.

Backward: holding the sampled record and the cache fixed, we remove only the last revision from the decoder input and replace only the earlier outputs. Completion drops by 1.7–2.3 points, so a single late revision changes how earlier chunks are rendered enough to change task outcomes. Raw drafts and rendered outputs show where the total gain arises. On Wan, VideoPhy-2 joint success rises from 25.5% (native) to 27.6% in AmendAR's raw drafts and to 33.2% after rendering, and StoryEval from 24.9% to 26.3% to 33.0%; MAGI and Hunyuan follow the same order. Most of the gain appears when earlier chunks are re-rendered, which is the step a native generator cannot take.

Table 4: Direct revision-use tests. Absolute completion means; VP-2, TC and Story denote VideoPhy-2, TC-Bench and StoryEval. Forward: drafts before decoding. Backward: fixed x/r and KV, last revision input removed and only earlier outputs replaced; the final chunk is retained. Bold compares each within-backbone pair. Backbone specifications are in Table 7.

Backbone	Readout	Completion (%)↑		
		VP-2	TC	Story
Forward path: raw drafts				
Wan	Full read	**27.6**	**18.5**	**26.3**
	r blocked	25.5	16.5	24.3
MAGI	Full read	**27.6**	**20.1**	**26.3**
	r blocked	24.3	17.5	23.9
Hunyuan	Full read	**19.5**	**13.5**	**20.2**
	r blocked	17.6	11.5	18.2
Backward path: earlier outputs				
Wan	Intact	**33.2**	**24.1**	**33.0**
	Last u off	31.2	22.0	30.7
MAGI	Intact	**31.8**	**24.4**	**31.9**
	Last u off	30.1	22.5	29.8
Hunyuan	Intact	**24.6**	**16.8**	**25.8**
	Last u off	22.7	15.1	24.0

The generator uses the revision history. To measure the forward term $\Delta(t)$ of Section 4.1 for the trained model, we fix the video KV states, targets, noise, and flow times and only block the revision keys. Flow-matching error rises by 3–7% on all backbones and benchmarks (Wan StoryEval: 0.828 → 0.873; Table 11), so the predictor depends on the explicit revisions beyond what the video states contain. Figure 15 in Appendix F maps this dependence over spatial

positions for one query. The backward path also reaches far: in $\sim$30 s videos, removing the last revision's contribution to the first chunk alone lowers subject consistency by 0.12–0.18 points and total scores by 0.06–0.08 (Table 6), and removing the revision at any single late position lowers all three task scores (Figure 6).

5.4 What limits the correction

Sampling the revision loses more than encoding it. Table 5 renders the same raw drafts in four ways: raw, the teacher target y, decoded encoder revisions $\tilde{r}$, and decoded sampled revisions r. Measured as a fraction of the teacher's gain over the raw draft, decoded $\tilde{r}$ keeps 84% or more on both endpoints and all backbones, while decoded r keeps 58–88%; on TC-Bench, sampled revisions trail encoded ones by 1.1–2.2 points. The codec thus represents most of what the teacher adds, and the main loss occurs when the generator has to produce the revision from the record, without the teacher's view of the whole video (Figure 17 resolves this across thresholds). The encoded revision of chunk i is computed from $y^{\leq i}$, which the teacher derived from the whole video, whereas the sampled revision is written at step i from $\mathcal{H}^{<i}$ and x^i; the gap measures how much of the teacher's correction the generator cannot yet anticipate when it writes the revision.

Table 5: Same-draft correction stages. Three-batch mean completion (%). All stages share the draft x. The teacher y and encoded $\tilde{r}$ use target information unavailable at deployment; r is sampled. Bold compares stages descriptively within each backbone.

Output	VP-2 joint↑	TC comp.↑
Wan2.1 / Self Forcing, 1.3B		
Raw draft x	27.6	18.5
Teacher y	**34.9**	**27.1**
Decode $\tilde{r}$	**34.9**	26.3
Decode r	33.2	24.1
MAGI-1, 4.5B-distill		
Raw draft x	27.6	20.1
Teacher y	**34.9**	**26.0**
Decode $\tilde{r}$	33.7	25.5
Decode r	31.8	24.4
HunyuanVideo 1.5 / minWM, 8.3B		
Raw draft x	19.5	13.5
Teacher y	25.3	**18.9**
Decode $\tilde{r}$	**25.8**	18.4
Decode r	24.6	16.8

State size limits what crosses chunk boundaries. Following Proposition 2, we refit decoders that transmit 128, 256, or 512 channels per state token while keeping the cell's local width and parameter count fixed. On Wan, TC-Bench completion with sampled revisions rises from 20.3% to 22.4% to 24.1%, encoded revisions reach 26.3% at full width, and MAGI shows the same trend (Figure 10). Transmission capacity therefore binds at small widths, while the gap between encoded and sampled revisions at full width is a separate limit set by the sampler. Because the state lives only in the small decoder, enlarging it leaves the generator and its KV cache unchanged.

Targets must follow the current model. Training on drafts from the initial generator instead of the current one lowers rendered TC-Bench completion by 2.6, 2.3, and 1.5 points on Wan, MAGI, and Hunyuan (Table 12); about 1.3–1.6 points of this difference already appear in the raw drafts. Refreshed targets describe the errors the current generator makes, which is what the revision is meant to correct.

Other video properties stay at the backbone's level. The remaining quality metrics move little: Hunyuan's consistent-attribute score falls from 58.5 to 58.0 and Wan's UMT-FVD changes from 161.2 to 163.8 (lower is better), while no single Wan system leads all quality and motion columns (Table 10). The gains are therefore specific to event completion, which is the property that depends on revising earlier chunks.

6 Conclusion

Chunk-wise video generators attend to and display the same latent, so every chunk is final once written. AmendAR separates the two: the cache stays append-only, and the displayed frames are decoded from the whole record. A small revision latent sampled after each chunk lets later chunks follow corrections and lets a backward decoder change earlier chunks, which raises event completion at a moderate inference cost. Because nothing in the cache is rewritten, the same design applies to three backbones that differ in size, sampler, step count, and attention layout. The same-draft comparison shows that the codec already keeps most of the teacher's gain, which makes the conditional sampling of revisions the most direct place to improve the method further.

REFERENCES

Hritik Bansal, Clark Peng, Yonatan Bitton, Roman Goldenberg, Aditya Grover, and Kai-Wei Chang. VideoPhy-2: A Challenging Action-Centric Physical Commonsense Evaluation in Video Generation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2503.06800.

Shuo Chen, Cong Wei, Sun Sun, Tiancheng Shen, Ping Nie, Kai Zou, Ge Zhang, Ming-Hsuan Yang, and Wenhu Chen. Context Forcing: Consistent Autoregressive Video Generation with Long Context. In *ICML*, 2026. URL https://icml.cc/virtual/2026/poster/61892.

Cheng Cheng, Lin Song, Di An, Yicheng Xiao, Xuchong Zhang, Hongbin Sun, and Ying Shan. From Prediction to Perfection: Introducing Refinement to Autoregressive Image Generation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2505.16324.

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 *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.02283.

Weixi Feng, Jiachen Li, Michael Saxon, Tsu-Jui Fu, Wenhu Chen, and William Yang Wang. TC-bench: Benchmarking temporal compositionality in conditional video generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4638–4662. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.241. URL https://aclanthology.org/2025.findings-acl.241/.

Jingwen He, Tianfan Xue, Dongyang Liu, Xinqi Lin, Peng Gao, Dahua Lin, Yu Qiao, Wanli Ouyang, and Ziwei Liu. VEnhancer: Generative Space-Time Enhancement for Video Generation. *arXiv preprint arXiv:2407.07667*, 2024. URL https://arxiv.org/abs/2407.07667.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the Train-Test Gap in Autoregressive Video Diffusion. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2506.08009.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive Benchmark Suite for Video Generative Models. In *CVPR*, pp. 21807–21818, 2024. URL https://arxiv.org/abs/2311.17982.

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*, 48(3):3268–3285, 2026. doi: 10.1109/TPAMI.2025.3633890. URL https://doi.org/10.1109/TPAMI.2025.3633890.

Xianghao Kong, Rob Brekelmans, and Greg Ver Steeg. Information-Theoretic Diffusion. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2302.03792.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling Forcing: Autoregressive Long Video Diffusion in Real Time. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25161.

Sand.ai, Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, W. Q. Zhang, Weifeng Luo, Xiaoyang Kang, Yuchen Sun, Yue Cao, Yunpeng Huang, Yutong Lin, Yuxin Fang, Zewei Tao, Zheng Zhang, Zhongshu Wang, Zixun Liu, Dai

Shi, Guoli Su, Hanwen Sun, Hong Pan, Jie Wang, Jiexin Sheng, Min Cui, Min Hu, Ming Yan, Shucheng Yin, Siran Zhang, Tingting Liu, Xianping Yin, Xiaoyu Yang, Xin Song, Xuan Hu, Yankai Zhang, and Yuqiao Li. MAGI-1: Autoregressive Video Generation at Scale. *arXiv preprint arXiv:2505.13211*, 2025. URL https://arxiv.org/abs/2505.13211.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2V-CompBench: A Comprehensive Benchmark for Compositional Text-to-video Generation. In *CVPR*, pp. 8406–8416, 2025. URL https://arxiv.org/abs/2407.14505.

Wan Team, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and Advanced Large-Scale Video Generative Models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Jiamian Wang, Ziqi Zhou, Chaithanya Kumar Mummadi, Sohail Dianat, Majid Rabbani, Raghuveer Rao, Chen Qiu, and Zhiqiang Tao. Visual self-refinement for autoregressive models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 21291–21300. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-emnlp.1161. URL https://aclanthology.org/2025.findings-emnlp.1161/.

Wenhao Wang and Yi Yang. VidProM: A Million-scale Real Prompt-Gallery Dataset for Text-to-Video Diffusion Models. In *NeurIPS*, volume 37, pp. 65618–65642, 2024. URL https://arxiv.org/abs/2403.06098.

Yiping Wang, Xuehai He, Kuan Wang, Luyao Ma, Jianwei Yang, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Is Your World Simulator a Good Story Presenter? A Consecutive Events-Based Benchmark for Future Long Video Generation. In *CVPR*, pp. 13629–13638, 2025b. URL https://arxiv.org/abs/2412.16211.

Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, Linus, Patrol, Peizhen Zhang, Peng Chen, Penghao Zhao, Qi Tian, Songtao Liu, Weijie Kong, Weiyan Wang, Xiao He, Xin Li, Xinchi Deng, Xuefei Zhe, Yang Li, Yanxin Long, Yuanbo Peng, Yue Wu, Yuhong Liu, Zhenyu Wang, Zuozhuo Dai, Bo Peng, Coopers Li, Gu Gong, Guojian Xiao, Jiahe Tian, Jiaxin Lin, Jie Liu, Jihong Zhang, Jiesong Lian, Kaihang Pan, Lei Wang, Lin Niu, Mingtao Chen, Mingyang Chen, Mingzhe Zheng, Miles Yang, Qiangqiang Hu, Qi Yang, Qiuyong Xiao, Runzhou Wu, Ryan Xu, Rui Yuan, Shanshan Sang, Shisheng Huang, Siruis Gong, Shuo Huang, Weiting Guo, Xiang Yuan, Xiaojia Chen, Xiawei Hu, Wenzhi Sun, Xiele Wu, Xianshun Ren, Xiaoyan Yuan, Xiaoyue Mi, Yepeng Zhang, Yifu Sun, Yiting Lu, Yitong Li, You Huang, Yu Tang, Yixuan Li, Yuhang Deng, Yuan Zhou, Zhichao Hu, Zhiguang Liu, Zhihe Yang, Zilin Yang, Zhenzhi Lu, Zixiang Zhou, and Zhao Zhong. HunyuanVideo 1.5 Technical Report. *arXiv preprint arXiv:2511.18870*, 2025. URL https://arxiv.org/abs/2511.18870.

Xunzhi Xiang, Zixuan Duan, Guiyu Zhang, Haiyu Zhang, Zhe Gao, Junta Wu, Shaofeng Zhang, Tengfei Wang, Qi Fan, and Chunchao Guo. Pathwise Test-Time Correction for Autoregressive Long Video Generation. In *ECCV*, 2026. URL https://arxiv.org/abs/2602.05871.

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

Jung Yi, Wooseok Jang, Paul Hyunbin Cho, Jisu Nam, Heeji Yoon, and Seungryong Kim. Deep Forcing: Training-Free Long Video Generation with Deep Sink and Participative Compression. In *ICML*, 2026. URL https://arxiv.org/abs/2512.05081.

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*, pp. 22963–22974, 2025. URL https://arxiv.org/abs/2412.07772.

Shenghai Yuan, Jinfa Huang, Yongqi Xu, Yaoyang Liu, Shaofeng Zhang, Yujun Shi, Ruijie Zhu, Xinhua Cheng, Jiebo Luo, and Li Yuan. ChronoMagic-Bench: A Benchmark for Metamorphic Evaluation of Text-to-Time-lapse Video Generation. In *NeurIPS*, volume 37, pp. 21236–21270, 2024. URL https://arxiv.org/abs/2406.18522.

Min Zhao, Hongzhou Zhu, Bokai Yan, Zihan Zhou, Yimin Chen, Wenqiang Sun, Kaiwen Zheng, Guande He, Xiao Yang, Chongxuan Li, Fan Bao, and Jun Zhu. minWM: A Full-Stack Open-Source Framework for Real-Time Interactive Video World Models. *arXiv preprint arXiv:2605.30263*, 2026. URL https://arxiv.org/abs/2605.30263v1.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal Forcing: Autoregressive Diffusion Distillation Done Right for High-Quality Real-Time Interactive Video Generation. In *ICML*, 2026. URL https://arxiv.org/abs/2602.02214.

AI Use Statement

Generative AI assisted with literature retrieval and analysis, mathematical derivations and proof checking, statistical consistency checks, figure preparation, and drafting and polishing paper sections. The authors take responsibility for all content and conclusions.

Overview of the Appendix

The appendix follows the argument of the paper. The main text makes three claims: a chunk that is finalized when it is written loses information that later chunks carry about it; appending a revision latent and decoding the displayed frames from the whole record recovers this information without touching the KV cache; and the recovered correction is limited by how well the generator samples revisions and by how much the decoder state can carry. Appendix A–C prove the statements behind these claims and measure each mechanism next to its derivation. Appendix A quantifies what finalizing early loses and measures how much later chunks change earlier chunks, Appendix B quantifies what revisions give to later chunks, and Appendix C shows why appending keeps the cache exact, what the decoder state limits, and what crosses a chunk boundary in practice. Appendix D gives the complete algorithm and every implementation choice together with its reason, and Appendix E gives the evaluation protocol and the cost. Appendix F reports the remaining results, grouped by the questions of Section 5.

A What finalizing a chunk early loses

Theorem 1 is the reason AmendAR exists: it states how much a chunk-wise generator loses by fixing the frames of chunk i at the step at which chunk i is written. This section proves the theorem, splits the error of the actual decoder into two parts, and then measures on trained models how much later chunks change earlier chunks (Appendix A.4).

A.1 Setting

All random variables live on a common probability space, and conditioning variables take values in standard Borel spaces, so regular conditional distributions exist. The target $y^i \in \mathbb{R}^{d_i}$ is the enhanced version of chunk i that training uses (Appendix D.3), and all estimators are square integrable. The joint law of $(c, x^{1:N}, r^{1:N}, y^{1:N})$ is fixed throughout each comparison. The record grows in two steps per chunk, first by the video latent and then by the revision latent:

$$\mathcal{G}_0 = \sigma(c), \qquad \mathcal{G}_{k,x} = \sigma(\mathcal{G}_{k-1}, x^k), \qquad \mathcal{G}_k = \sigma(\mathcal{G}_{k,x}, r^k). \tag{6}$$

Thus $\mathcal{G}_{k-1} \subseteq \mathcal{G}_{k,x} \subseteq \mathcal{G}_k$, and no independence between chunks is assumed. The best estimates of y^i after the two steps are $\mu^i_{k,x} = \mathbb{E}[y^i \mid \mathcal{G}_{k,x}]$ and $\mu^i_k = \mathbb{E}[y^i \mid \mathcal{G}_k]$, and the Bayes risk after chunk k is $\mathcal{B}^i(k) = \mathbb{E}\|y^i - \mu^i_k\|^2$. A native generator corresponds to empty revisions, so that $\mathcal{G}_{k,x} = \mathcal{G}_k$.

Lemma 3 (Orthogonal projection). *For square-integrable z and a square-integrable $\mathcal{F}$-measurable estimate g,*

$$\mathbb{E}\|z - g\|^2 = \mathbb{E}\|z - \mathbb{E}[z \mid \mathcal{F}]\|^2 + \mathbb{E}\|g - \mathbb{E}[z \mid \mathcal{F}]\|^2. \tag{7}$$

If $\mathcal{F} \subseteq \mathcal{F}'$, then

$$\mathbb{E}\|z - \mathbb{E}[z \mid \mathcal{F}]\|^2 - \mathbb{E}\|z - \mathbb{E}[z \mid \mathcal{F}']\|^2 = \mathbb{E}\|\mathbb{E}[z \mid \mathcal{F}'] - \mathbb{E}[z \mid \mathcal{F}]\|^2. \tag{8}$$

Proof. Write $z - g = (z - \mathbb{E}[z \mid \mathcal{F}]) + (\mathbb{E}[z \mid \mathcal{F}] - g)$. The second term is $\mathcal{F}$-measurable and the first has zero conditional mean given $\mathcal{F}$, so the expected inner product vanishes; square integrability makes the product integrable. This proves Eq. (7). Applying it with the larger σ-algebra $\mathcal{F}'$ and the $\mathcal{F}'$-measurable estimate $\mathbb{E}[z \mid \mathcal{F}]$ proves Eq. (8). The argument uses neither densities nor Gaussianity. □

A.2 Proof of Theorem 1

The contributions of the video and the revision latent of chunk k to the estimate of y^i are

$$V^i_{k,x} = \mathbb{E}\|\mu^i_{k,x} - \mu^i_{k-1}\|^2, \qquad V^i_{k,r} = \mathbb{E}\|\mu^i_k - \mu^i_{k,x}\|^2. \tag{9}$$

Applying Lemma 3 to both inclusions in Eq. (6), the Bayes risk falls by $V^i_{k,x} + V^i_{k,r}$ from $\mathcal{G}_{k-1}$ to $\mathcal{G}_k$. Summing over $k = s+1, \dots, N$ telescopes:

$$\mathcal{B}^i(s) - \mathcal{B}^i(N) = \sum_{k=s+1}^{N} \big(V^i_{k,x} + V^i_{k,r}\big). \tag{10}$$

For a $\mathcal{G}_s$-measurable estimate $\hat{y}^i_{\mathrm{c}}$ and a $\mathcal{G}_N$-measurable estimate $\hat{y}^i_{\mathrm{a}}$, a second application of Eq. (7) gives $\mathcal{R}^i_{\mathrm{c}} = \mathcal{B}^i(s) + \mathcal{E}^i_{\mathrm{c}}$ and $\mathcal{R}^i_{\mathrm{a}} = \mathcal{B}^i(N) + \mathcal{E}^i_{\mathrm{a}}$. Subtracting proves Eq. (1). □

Reading the theorem on a concrete prompt. Take "a man picks an apple and places it in a basket" and let y^1 be the enhanced first chunk. If the basket first becomes visible in chunk 4, the latent x^4 changes the best estimate of where the hand in chunk 1 should reach, so $V^1_{4,x} > 0$. A revision r^4 sampled after x^4 can state this change directly, which adds $V^1_{4,r}$. A native generator finalizes chunk 1 at $s = 1$ and loses both terms for every $k \geq 2$; a rolling window of w chunks finalizes it at $s = w$ and still loses every term with $k > w$. Chunks that leave the best estimate unchanged contribute zero, so the sum is large exactly for prompts whose early setup depends on later events, which are the prompts that the completion benchmarks contain.

What the theorem requires of a method. Realizing the sum takes three things, and each corresponds to one component of AmendAR. The record must contain information about earlier chunks; the revision latent adds the terms $V^i_{k,r}$ on top of what video latents carry. An estimator must read the record after step i; the backward decoder does this for every earlier chunk. The fitting error $\mathcal{E}^i_{\mathrm{a}}$ must stay below the information gain, since a poorly fitted decoder can increase risk even when useful information arrives; this requires targets that are well defined for each draft, which is why y^i enhances the draft itself.

A.3 What the decoder can use

The decoder in Eq. (3) does not read all of $\mathcal{G}_N$ when it renders chunk i. Its inputs generate $\mathcal{D}^i = \sigma(x^i, q^i, \dots, q^N, u^i, \dots, u^N) \subseteq \mathcal{G}_N$; it has no direct input for c or $x^{<i}$, although their information can reach it indirectly through the sampled later chunks. Let $\mu^i_{\mathcal{D}} = \mathbb{E}[y^i \mid \mathcal{D}^i]$. Because $\hat{y}^i_{\mathrm{a}} = \hat{x}^{i|N}$ is $\mathcal{D}^i$-measurable and the tower property gives $\mathbb{E}[\mu^i_N \mid \mathcal{D}^i] = \mu^i_{\mathcal{D}}$,

$$\mathcal{E}^i_{\mathrm{a}} = \underbrace{\mathbb{E}\|\mu^i_N - \mu^i_{\mathcal{D}}\|^2}_{\text{information the decoder cannot see}} + \underbrace{\mathbb{E}\|\hat{y}^i_{\mathrm{a}} - \mu^i_{\mathcal{D}}\|^2}_{\text{error within the information it sees}}. \tag{11}$$

The cross term vanishes after conditioning on $\mathcal{D}^i$. The first term remains for any decoder of these inputs, however large; the second contains architectural restrictions, the finite transmitted state, optimization, and estimation error.

How the experiments probe each term. The decomposition tells which experiment isolates which loss, and Appendix F follows this map. Setting the incoming state $h^{i+1} = 0$ shrinks $\mathcal{D}^i$ to the local

inputs (x^i, u^i) and so enlarges the first term; the drop it causes (Table 3) measures what later chunks contribute. Restricting the transmitted state to fewer channels leaves $\mathcal{D}^i$ unchanged in principle but limits the second term through the rank bound of Proposition 2; the width study measures this limit. Replacing sampled revisions r by encoded revisions $\tilde{r}$ changes the joint law of the inputs, because $\tilde{r}$ is computed from the teacher's view of the whole video, and the same-draft comparison measures how much of the teacher's correction the sampled revisions lose.

A.4 What later chunks change in earlier chunks

Theorem 1 predicts a large loss from finalizing early exactly when later chunks move the best estimate of earlier chunks. The following measurements check this on the trained models, from the decoder state to the task scores and the rendered frames.

The decoder state of an early chunk moves as later chunks arrive. The sum in Eq. (1) is nonzero only if later chunks change what the model knows about an earlier chunk. Figure 5 follows the decoder state h^i of one early position during ordinary streaming, as the prefix grows from $k = 1$ to $k = 7$. With a prefix of one chunk, the state of an early position shows little structure, because it has seen only local content. Once later chunks are available (prefix 3), it acquires structure aligned with the bowl and the hand, and it changes less between prefixes 5 and 7 than between prefixes 1 and 3, so most of what later chunks say about this early chunk arrives within a few chunks.

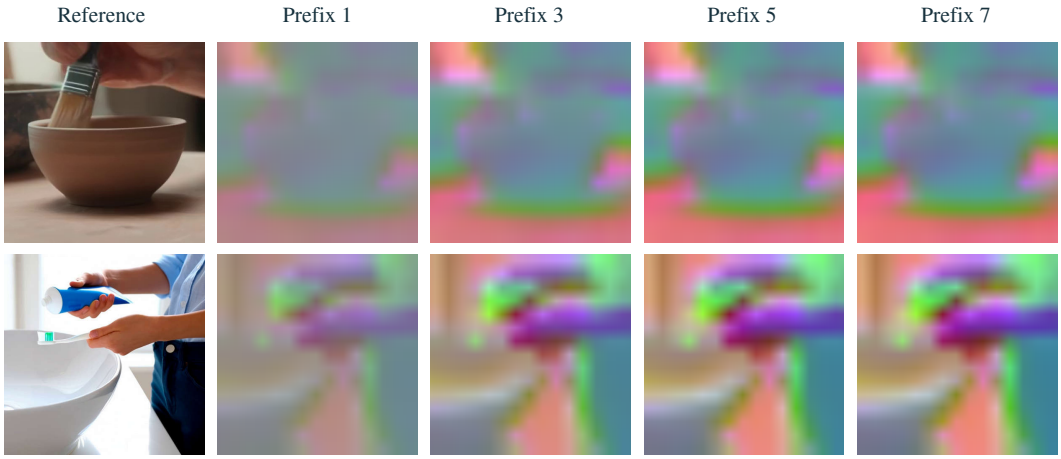


Figure 5: An early decoder state changes as the prefix grows. VideoPhy-2 pottery coating (top) and toothpaste dispensing (bottom). Reference views accompany the state h^i at one fixed early position as the prefix grows to $k = 1$, 3, 5, and 7 chunks. One PCA fitted to all eight states fixes the center, component directions, and per-component 99.5th-percentile absolute scales, and RGB encodes three projected coordinates.

Every late revision changes earlier outputs. Figure 6 applies the fixed-record intervention of Appendix D.5 at each late position j separately and reports the retained fraction of each task score. Every single removal lowers all three scores on all backbones, so the gain does not rest on one particular revision. Removing the last revision costs the most in all nine curves: on Wan, 6–9% of each score is lost, against 2–6% for position 2. Two factors grow with j. A later revision is sampled after more of the video exists, so it can carry more information about earlier chunks, and removing it replaces more earlier outputs, all $i < j$. The sweep does not separate the two factors, but the curves are not monotone: on Wan, removing position 5 costs less than removing position 4 although it replaces one more output. The size of the effect therefore depends on what a revision contains and on more than the number of outputs it reaches.

The change appears where the event happens. Figure 7 pairs two raw drafts with their rendered outputs from the same record. In the clay-pot draft, the pot lands intact and breaks one stage later; in the rendered output it breaks at impact, as the prompt describes. In the frisbee draft, the frisbee flies higher in the boxed stages, while the rendered output brings it closer to the ground, where the prompt says it should tumble. In both cases the rendering changes the stages in which the event happens and leaves the scene, the person, and the camera unchanged.

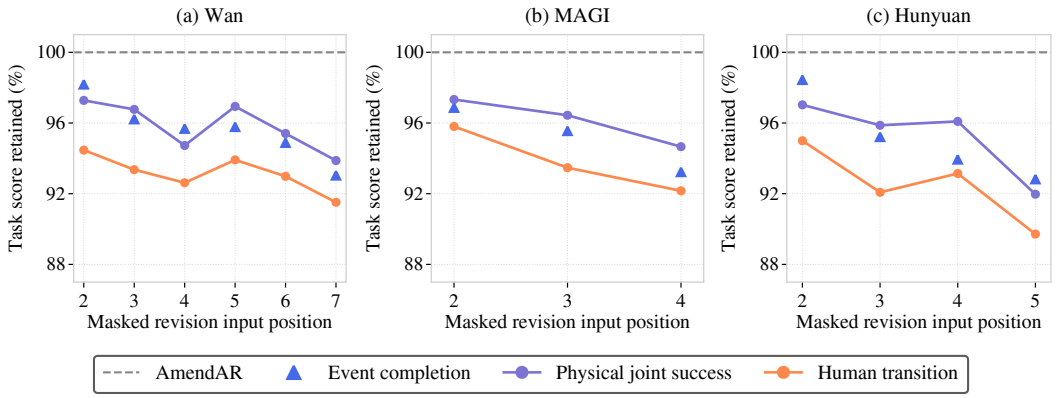


Figure 6: Later revisions change earlier outputs under a fixed record. At position j, we set the decoder input $u^j = 0$, keep the incoming state h^{j+1}, and recompute the backward scan, replacing only the outputs $i < j$. The stored x/r record, the KV cache, and the outputs $i \geq j$ stay fixed. Each position is removed separately from the original record; values average $100\,\mathrm{score}_{\mathrm{removed}}/\mathrm{score}_{\mathrm{AmendAR}}$ over three seeds. Per seed, VideoPhy-2 evaluates 590 videos (both scores ≥ 4), TC-Bench 750 videos (mean transition rating $\geq 11/3$), and StoryEval 423 prompts (fraction of events completed). Positions index the removed input.

The longest connection still acts, on appearance. Table 6 tests the connection from the last revision to the first chunk in ∼30 s videos, across 30–40 chunks. Keeping it raises subject consistency by 0.12–0.18 points and total scores by 0.06–0.08, while dynamic degree changes by −0.05 to +0.02 points. The first chunk thus keeps the subject's appearance closer to its later appearance without a change in the amount of motion, which is the kind of correction a late revision can supply about an early chunk: how the subject looks once more of it has been seen. The effect of one revision over this distance is small, as the decay in Figure 9 predicts.

Table 6: Last-to-first intervention at ∼30 s. Intact-minus-removed-edge means in 10^{-2} VBench-Long points. Only the first chunk's output latent is replaced before full VAE decoding. Rows are distinct native operating points, not a cross-backbone ranking. Backbone specifications are in Table 7.

Backbone	Δ Subj.	Δ Total	Δ Dyn.
Wan	17.6	8.1	-0.6
MAGI	12.4	6.0	1.9
Hunyuan	13.3	6.7	-4.9

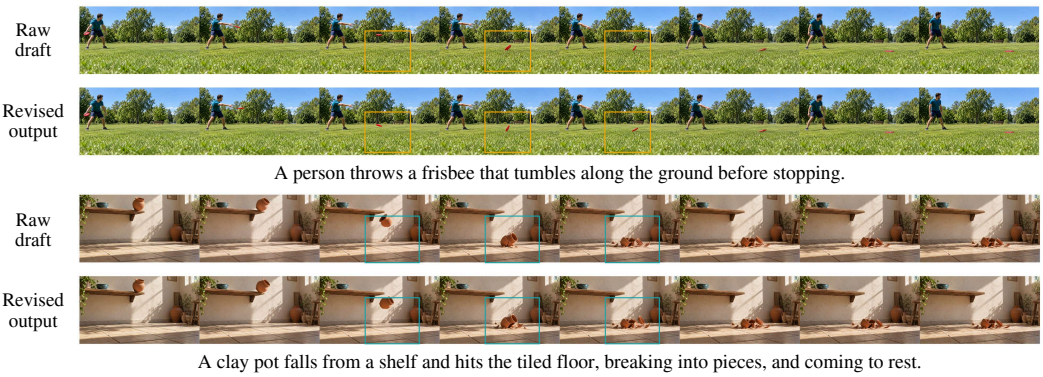


Figure 7: Raw drafts and revised outputs from the same generation history. Each case pairs its raw draft x (upper row) with its rendered output $\hat{x}$ (lower row) at eight corresponding stages. The upper case shows a frisbee tumbling along the ground, and the lower case shows a clay pot falling and breaking. The paired rows display the progression of each event before and after revision.

B What revisions give to later chunks

The decoder uses revisions to change earlier chunks. The same revisions also sit in the KV cache, where later video chunks attend to them. This forward use keeps later chunks consistent with the corrected earlier chunks the viewer sees, and Section 4.1 states how much it can help through the loss

reduction $\Delta(t)$. This section derives $\Delta(t)$, shows when it is zero, relates it to mutual information, and explains why the forward and the backward path must be tested separately.

B.1 Optimal flow-matching loss with and without revision access

Fix a square-integrable future chunk $z \in \mathbb{R}^d$. Let $\mathcal{Q}$ denote the explicitly readable revision history and $\mathcal{X}$ everything retained after that access is removed, including the unchanged video KV states; revision information already encoded in those video states stays in $\mathcal{X}$. Let $\epsilon \sim \mathcal{N}(0, I_d)$ be independent of $(z, \mathcal{X}, \mathcal{Q})$, $z_t = tz + (1-t)\epsilon$, and $w = z - \epsilon$ the target velocity. For fixed $t \in (0,1)$, the optimal losses over all square-integrable measurable predictors are

$$L^\star_{\text{drop}} = \inf_g \mathbb{E}\|w - g(z_t, \mathcal{X})\|^2, \qquad L^\star_{\text{full}} = \inf_g \mathbb{E}\|w - g(z_t, \mathcal{X}, \mathcal{Q})\|^2. \tag{12}$$

Their minimizers are the conditional means of w. Since $w = (z - z_t)/(1-t)$, the two means differ by $(\mathbb{E}[z \mid z_t, \mathcal{X}, \mathcal{Q}] - \mathbb{E}[z \mid z_t, \mathcal{X}])/(1-t)$, and Lemma 3 applied to the nested inputs gives

$$L^\star_{\text{drop}} - L^\star_{\text{full}} = \Delta(t) = \frac{\mathbb{E}\big\|\mathbb{E}[z \mid z_t, \mathcal{X}, \mathcal{Q}] - \mathbb{E}[z \mid z_t, \mathcal{X}]\big\|^2}{(1-t)^2}, \tag{13}$$

the quantity stated in Section 4.1. The identity concerns optimal predictors; element-averaged losses divide both sides by d. The numerator measures how much the best prediction of the future chunk moves when the explicit revisions are added, and the factor $(1-t)^{-2}$ converts this change of the data prediction into a change of the velocity loss.

Proposition 4 (Zero loss gap and conditional independence)**.** *Under the preceding assumptions, $\Delta(t) = 0$ for any one fixed $t \in (0,1)$ if and only if z is conditionally independent of $\mathcal{Q}$ given $\mathcal{X}$.*

Proof. Conditional independence makes the two posterior means equal. Conversely, $\Delta(t) = 0$ makes them equal almost surely. For almost every conditioning value $(\chi, \varkappa)$ of $(\mathcal{X}, \mathcal{Q})$, Gaussian convolution gives z_t strictly positive smooth densities. With $\varsigma = 1 - t$, differentiating under the Gaussian integral gives

$$\nabla_v \log p(v \mid \chi, \varkappa) = \frac{t\,\mathbb{E}[z \mid z_t = v, \chi, \varkappa] - v}{\varsigma^2}, \qquad \nabla_v \log p(v \mid \chi) = \frac{t\,\mathbb{E}[z \mid z_t = v, \chi] - v}{\varsigma^2}. \tag{14}$$

Positivity turns almost-sure equality into equality for Lebesgue-almost every v, and smoothness extends it to every v. The log densities then differ by a constant on the connected space $\mathbb{R}^d$, and normalization makes the constant zero. The characteristic functions satisfy $\varphi_{z_t|\chi,\varkappa}(\omega) = \varphi_{z|\chi,\varkappa}(t\omega)\exp(-\varsigma^2\|\omega\|^2/2)$, and the Gaussian factor never vanishes, so the conditional laws of z given $(\chi, \varkappa)$ and given χ coincide. The argument needs $0 < t < 1$, which is why the endpoints are excluded. □

B.2 An information integral and its regularity conditions

Proposition 4 states when $\Delta(t)$ vanishes at one flow time. A weighted integral of $\Delta(t)$ over flow times equals the conditional mutual information between the future chunk and the revision history, so the loss gap is an information quantity. Write $\mathcal{I}(z; \mathcal{Q} \mid \mathcal{X})$ for conditional mutual information in nats. The Gaussian information identity used in diffusion analysis (Kong et al., 2023) gives

$$\mathcal{I}(z; \mathcal{Q} \mid \mathcal{X}) = \int_0^1 \frac{t}{1-t}\,\Delta(t)\,\mathrm{d}t. \tag{15}$$

The derivation below applies when the Gaussian-smoothed conditional densities permit differentiating entropy and integrating by parts with vanishing boundary terms; bounded z suffices, and the identity extends through square-integrable truncation when the integrated quantities converge. The integral may be infinite. These conditions are needed only here, not for Eq. (13) or Proposition 4.

Proof. Let $\mathcal{S}$ be an arbitrary conditioning variable, $\gamma > 0$ a signal-to-noise ratio, $W_\gamma = \sqrt{\gamma}\,z + \epsilon$, and $V_\lambda = z + \sqrt{\lambda}\,\epsilon$ for variance $\lambda > 0$. The density p_λ of V_λ solves $\partial_\lambda p_\lambda = \frac{1}{2}\nabla^2 p_\lambda$. Differentiating $h(p_\lambda) = -\int p_\lambda \log p_\lambda$ and integrating by parts gives $\mathrm{d}h(p_\lambda)/\mathrm{d}\lambda = \frac{1}{2}J(p_\lambda)$, where $J(p_\lambda) = \int \|\nabla \log p_\lambda\|^2 p_\lambda$ is the Fisher information. The score of V_λ gives

$$J(V_\lambda \mid \mathcal{S}) = \frac{d}{\lambda} - \frac{1}{\lambda^2}\,\mathbb{E}\big\|z - \mathbb{E}[z \mid V_\lambda, \mathcal{S}]\big\|^2. \tag{16}$$

To see the second term, project the noise $\sqrt{\lambda}\,\epsilon = V_\lambda - z$ onto $(V_\lambda, \mathcal{S})$: its total squared norm is $d\lambda$, and its residual equals the negative estimation residual of z; Lemma 3 completes the step. With $\lambda = 1/\gamma$ and $h(W_\gamma \mid \mathcal{S}) = h(V_{1/\gamma} \mid \mathcal{S}) + \frac{d}{2}\log\gamma$, differentiation yields

$$\frac{\mathrm{d}}{\mathrm{d}\gamma}\,\mathcal{I}(z; W_\gamma \mid \mathcal{S}) = \frac{1}{2}\,\mathrm{mmse}(z \mid W_\gamma, \mathcal{S}). \tag{17}$$

Subtract the identities for $\mathcal{S} = \mathcal{X}$ and $\mathcal{S} = (\mathcal{X}, \mathcal{Q})$. Since W_γ and $\mathcal{Q}$ are conditionally independent given $(z, \mathcal{X})$, the difference of information terms equals $\mathcal{I}(\mathcal{Q}; W_\gamma \mid \mathcal{X})$. It is zero at $\gamma = 0$ and tends to $\mathcal{I}(\mathcal{Q}; z \mid \mathcal{X})$ as $\gamma \to \infty$, which follows by recovering finite continuity-set quantizations of z from $W_\gamma/\sqrt{\gamma}$ and approximating mutual information by finite quantizations; the nonnegative MMSE difference permits an infinite limit. Finally, $\gamma = t^2/(1-t)^2$ has derivative $2t/(1-t)^3$, and the MMSE difference equals $(1-t)^2\Delta(t)$ by Eq. (13), which gives Eq. (15). □

B.3 Trained predictors and the two paths

The identities above describe optimal predictors. The fixed-query test of Section 5.3 evaluates one trained network with and without revision keys, so its loss difference contains fitting errors as well. For trained predictors v_{drop} and v_{full}, let α_{drop} and α_{full} be their squared distances from the corresponding optimal velocities. The same projection identity gives

$$L(v_{\mathrm{drop}}) - L(v_{\mathrm{full}}) = \Delta(t) + \alpha_{\mathrm{drop}} - \alpha_{\mathrm{full}}. \tag{18}$$

One network evaluated with and without an attention edge has different fitting errors, so its loss change can take either sign and measures the network's dependence on the edge. Our fixed-query test keeps the target, noise, time, and video KV fixed, including any revision information already present there, and therefore measures dependence on the explicit revision access. The first chunk has no revision history and stays in the evaluation with zero access difference.

Why the two paths are tested separately. The forward and backward gains are logically independent, so a gain on one path says nothing about the other. Take independent fair bits b_1 and b_2 and a constant retained context. If a late revision reveals b_1, an earlier target equal to b_1 becomes reconstructible, while a future target equal to b_2 gains nothing. Conversely, if the earlier target is already available as a local input b_2 and the revision history reveals b_1, the revision adds nothing to reconstructing the earlier target but predicts a future target equal to b_1. Each path must therefore be tested on its own, as in Table 4.

C Why appending keeps the cache exact, and what the decoder state limits

AmendAR changes earlier frames without changing any stored latent. This section makes the consequences precise. Appending leaves every stored KV state exactly as it was (Appendix C.1), whereas overwriting an old latent invalidates a large part of the cache (Appendix C.2). The backward decoder, which carries the correction, passes it through a fixed-size state, and this size bounds the number of correction directions that can reach an earlier chunk (Appendix C.3–C.4), and Appendix C.5 measures what crosses a boundary in the trained decoder. Appendix C.6 counts the added computation and checks the count against the measured cost.

C.1 Proof of cache invariance

Consider deterministic evaluation of a Transformer, or hold all internal randomness fixed. For every stored row, hold its input embedding, position, clean-pass time condition, text condition, and attention-visible set fixed. Positionwise operations act on each row separately, so at layer zero the stored hidden vectors are unchanged when new rows are appended. Suppose they are unchanged through layer $\ell - 1$. Their key and value projections at layer ℓ are then unchanged, every stored query reads the same keys and values under the same position encoding and mask, and its attention weights and output are unchanged. Residual updates, normalization, and feed-forward layers preserve equality. By induction, all stored hidden vectors and KV states agree at every layer, so evaluating only the new query rows gives the same result as recomputing the stored rows. This proves the first part of Proposition 2.

C.2 WHY OVERWRITING AN OLD LATENT INVALIDATES REUSE

An overwrite changes an input of the finished computation graph. View any cached node as a differentiable function $g(x^i)$ of an old latent x^i. If $Dg(x^i)[\delta] \neq 0$, then

$$g(x^i + \eta\delta) - g(x^i) = \eta\, Dg(x^i)[\delta] + o(|\eta|) \neq 0 \quad \text{for all sufficiently small } \eta \neq 0, \tag{19}$$

because the remainder has norm below $\frac{1}{2}|\eta|\|Dg(x^i)[\delta]\|$ for small $|\eta|$ and cannot cancel the linear term. Such a cached value cannot be reused in an exact evaluation of the overwritten input. Nodes outside the dependency graph are unaffected, and reachable nodes can have zero directional response, so an overwrite requires recomputing a dependent subgraph. In a causal Transformer this subgraph is large. On Wan, replacing x^1 after the seventh chunk leaves the first-layer keys and values of the later rows unchanged, because they depend only on each row's own input, but every later layer mixes x^1 into the hidden states of all 6×4680 later video rows, so their KV states in 29 of the 30 layers become stale. Recomputing them restores a valid cache, yet the later chunks themselves were sampled for the old content of x^1 and would still have to be resampled. The append-only design keeps every old input fixed and changes only the separate rendering computation.

C.3 A NECESSARY STATE DIMENSION

Collect the local inputs $e^l = (x^l, r^l)$ and the residual outputs $o^l = \hat{x}^{l|k} - x^l$. Fix a differentiable operating point and a boundary after chunk j. With $e^{\leq j}$ held fixed, every effect of $e^{>j}$ on $o^{1:j}$ passes through $h^{j+1} \in \mathbb{R}^m$, so $o^{1:j} = a_j\big(e^{\leq j}, b_j(e^{>j})\big)$ for differentiable maps a_j and b_j with $b_j(e^{>j}) = h^{j+1}$. The chain rule gives

$$J_j := \frac{\partial o^{1:j}}{\partial e^{>j}} = \frac{\partial a_j}{\partial h^{j+1}}\,\frac{\partial b_j}{\partial e^{>j}}, \qquad \operatorname{rank} J_j \leq m. \tag{20}$$

This proves the second part of Proposition 2. The bound binds in practice. On Wan, one chunk latent has $3 \times 60 \times 104 \times 16 = 299{,}520$ scalars, while the full state transmits $m = 390 \times 512 = 199{,}680$; on HunyuanVideo 1.5 the two sizes coincide at 199,680. A correction that changes every direction of an earlier chunk independently of the others therefore cannot cross even one chunk boundary on Wan, and the decoder has to learn which directions matter. Reproducing a target local response $J_j^\star$ therefore requires $m \geq \operatorname{rank} J_j^\star$ at every boundary, counting both later video and revision inputs. The decoder need not be linear, but the bound concerns a local derivative. A conditional mean defined only almost surely has no unique derivative outside its support, so any derivative-based comparison with a target must specify a differentiable extension or a neighborhood where the target function is identified.

Theorem 5 (Exact state size for a general linear reverse scan)**.** *Let $o^i = \sum_{l=i}^{N} M_{il}\, e^l$ be any block upper-triangular linear map. Among time-varying linear reverse scans that allow a direct local input in each output, the minimum uniform state dimension is*

$$m_\star = \max_{1 \leq j < N} \operatorname{rank} M_{1:j,\, j+1:N}, \tag{21}$$

with $m_\star = 0$ for $N = 1$.

Proof. Necessity follows from Eq. (20). For sufficiency, let $H_j = M_{1:j,\, j+1:N}$ and $m_j = \operatorname{rank} H_j$. Choose a basis embedding $B_j : \mathbb{R}^{m_j} \to \operatorname{im} H_j$ and its coordinate map C_j on that image, extended linearly to the output space. Define the state $h^{j+1} = C_j H_j e^{>j}$, so that $B_j h^{j+1}$ is exactly the contribution of all later inputs to outputs $1, \ldots, j$. Starting from $h^{N+1} = 0$, the current output and the previous state are recovered by

$$o^j = M_{jj}\, e^j + \Pi_j B_j h^{j+1}, \qquad h^j = C_{j-1}\big(M_{1:j-1,\, j}\, e^j + \Pi_{<j} B_j h^{j+1}\big), \tag{22}$$

where Π_j selects output block j and $\Pi_{<j}$ the preceding blocks. The expression in parentheses equals $H_{j-1} e^{\geq j}$, so the state keeps its meaning by backward induction. Empty end states have dimension zero, and padding every state to $m_\star$ gives a uniform realization. □

The theorem allows time-varying matrices and an unrestricted local readout. The AmendAR cell shares parameters across chunks and forms its residual through $P_o(h^i)$, which can add restrictions such as a limited local readout rank. The theorem shows that the transmitted-state bound is tight in this larger class.

C.4 Residual error from insufficient transmission

The rank bound states which responses are impossible. The following bound states how much error remains when the state is too small, measured on the local response to perturbations of the later inputs.

Fix the earlier inputs and differentiable target and model functions $F^\star, F_m$ of the later inputs near a point e. Let ξ be a zero-mean perturbation with finite support and covariance Σ, and let $H^\star = DF^\star(e)$ and $H_m = DF_m(e)$ with $\operatorname{rank} H_m \le m$. Then

$$\lim_{\eta\to 0} \eta^{-2}\,\mathbb{E}\big\|[F^\star(e+\eta\xi) - F^\star(e)] - [F_m(e+\eta\xi) - F_m(e)]\big\|^2 = \big\|(H^\star - H_m)\Sigma^{1/2}\big\|_F^2 \ge \sum_{l>m} \sigma_l^2\big(H^\star\Sigma^{1/2}\big). \tag{23}$$

For the equality, expand both functions at e, divide by η, and pass the finite expectation through the limit; the covariance converts the quadratic form to a Frobenius norm. For the inequality, set $A = H^\star\Sigma^{1/2}$ and $B = H_m\Sigma^{1/2}$, which has rank at most m. If P projects onto the column space of B, then $\|A-B\|_F^2 \ge \|(I-P)A\|_F^2$. In an eigenbasis of $AA^\top$, the diagonal entries of P lie in $[0,1]$ and sum to at most m, so $\operatorname{tr}(PAA^\top)$ is at most the sum of the m largest eigenvalues, which gives the singular-value tail. This bound describes local sensitivity around a specified target function; information the decoder cannot see contributes the separate term in Eq. (11).

C.5 What crosses a chunk boundary in the trained decoder

Proposition 2 bounds what the state can carry across a boundary. The following measurements show what it carries in the trained decoder, how the influence of one revision fades over several boundaries, and how the correction changes when the state is made smaller.

The change is local and decays with distance. Figure 9 removes u^7 and shows where four earlier chunks change. The response is strongest on the hands above the dough in the first example and on the lamp beside the paper crane in the second, so it concentrates on a few regions of the frame. It also grows from chunk 1 to chunk 4, the chunk closest to the removed revision. The information of u^7 reaches chunk i only through the states $h^7, \dots, h^{i+1}$, and each cell also mixes in its own local content, so the influence of one revision fades with the number of cells it passes through. Chunk 1 is six boundaries away from u^7, so what it still receives has passed through six applications of f_ϕ. The fading also explains why several late revisions, and not only the last one, contribute to the task scores in Figure 6: each revision acts most strongly on the chunks just before it.

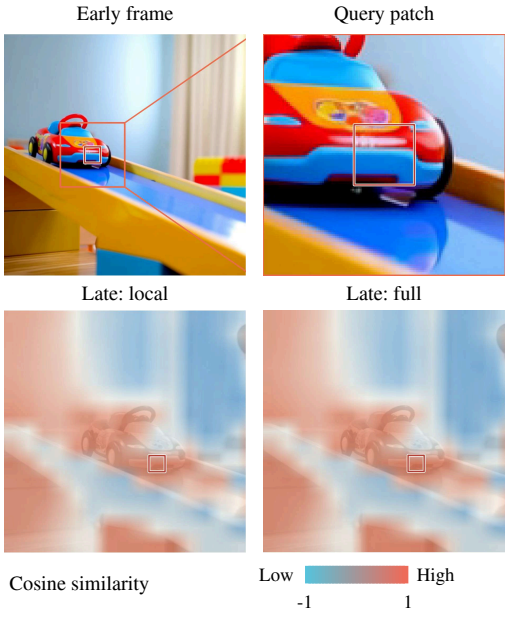


Figure 8: Early state and late correspondence. Early frame and query appear above the two late cosine fields; both conditions use the same query position.

The state links an early position to the object it becomes. Figure 8 examines one state vector in a VideoPhy-2 toy-car sequence, at 0.00 and 3.75 s (source frames 0 and 60 at 16 fps). Each condition takes its own early 512-dimensional state vector at the zero-based grid location $(7, 5)$ and compares it with the common final state. Both similarity fields peak at grid cell $(9, 9)$ on the later car. The early representation and its similarity field change with the reading condition, while the matched later position stays on the same object. The early state thus changes with what the decoder reads from later chunks and still refers to the same object, which is what a correction of that object in the early chunk requires.

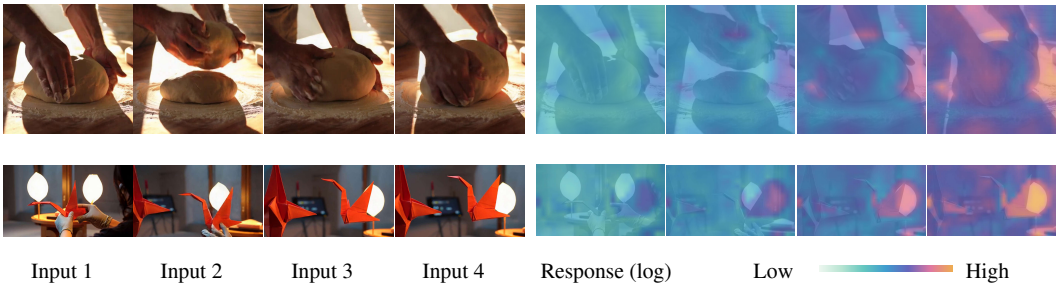
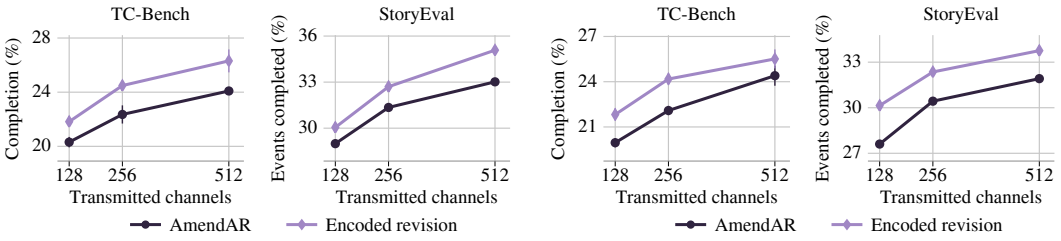


Figure 9: One late revision changes earlier chunks at specific locations. VideoPhy-2 dough shaping (top) and origami folding (bottom). Each row pairs four early reference views ($i = 1, \dots, 4$) with the RMS latent change $\|\hat{x}^i_{\text{full}} - \hat{x}^i_{u^7=0}\|/\sqrt{d_x}$ caused by removing u^7, where d_x is the latent channel count. The fixed-record protocol is in Appendix D.5. All eight 15×26 fields share the scale $\log(1 + 50\,\delta/0.0169810)/\log 51$ for change δ and 20% grayscale reference texture, so color locates the response and encodes its magnitude on one logarithmic scale.

State size and sampling are separate limits. Figure 10 varies the transmitted state at a fixed 512-channel local cell, with the encoder, the sampled record, the parameter count, and the initialization held fixed. Both revision sources improve at every doubling of the width, on both tasks and both backbones, and the curves are still rising at 512 channels, so the tested widths do not saturate the decoder. The gap between encoded and sampled revisions lies between about 1.1 and 2.7 points at every width and does not close as the state grows: it widens with width on Wan, narrows on MAGI, and is still 1.1–2.2 points at 512 channels. If sampled revisions fell short only because the state was too small, a larger state would close the gap; it does not, so sampling and state size limit the correction separately.



(a) Wan: AmendAR, fixed 512-channel local cell

(b) MAGI: AmendAR, fixed 512-channel local cell

Figure 10: Transmitted state size limits correction at fixed local cell size. TC-Bench completion and StoryEval event completion at 128, 256, and 512 transmitted channels, with a fixed 512-channel local cell. Dark curves decode sampled revisions r; light curves decode encoded revisions $\tilde{r}$. Points and bars show mean $\pm$ sample SD over three seeds. Each seed evaluates 750 TC-Bench videos (mean transition rating $\geq 11/3$) and 423 StoryEval prompts. All state tensors stay densely allocated; only the transmitted channels change.

Figure 11 shows what the extra channels carry. At 512 channels, the state of the earlier stage differs most from the local-only state over the lower slope, where the dust appears in the later stage. A wider state therefore lets later content reach the earlier chunk at the place where the later event happens.

C.6 Counting the additional work

Let each chunk contain n_x video and n_r revision tokens, $\rho = n_r/n_x$, and let the samplers use K_x and K_r steps; $P_x = K_x + 1$ and $P_r = K_r + 1$ include the clean passes. The native model processes $P_x N n_x$ query rows, and AmendAR adds $P_r N n_r$, an extra query-linear ratio of $(P_r/P_x)\rho$. Under dense full-history attention, native interactions number $P_x n_x^2 N(N+1)/2$. Video queries additionally read past revision keys, contributing $P_x n_x n_r N(N-1)/2$, and revision queries contribute $P_r n_r (n_x + n_r) N(N+1)/2$. The extra attention ratio is therefore

$$\rho \frac{N-1}{N+1} + \frac{P_r}{P_x}\rho(1+\rho), \tag{24}$$

whose first term vanishes at $N = 1$. With equal KV precision and width, persistent visual KV grows by the factor $1+\rho$; with L layers of KV width d_{kv} and b bytes per scalar, it occupies $2bLd_{\text{kv}}N(n_x + n_r)$ bytes. One final reverse scan uses N cell evaluations and transmits $(N-1)m$ state scalars, and rendering every prefix costs $\sum_{k=1}^{N} k = N(N+1)/2$ evaluations. These counts exclude hardware effects, text processing, VAE decoding, and target preparation; the timing protocol in Appendix E.3 measures practical cost.

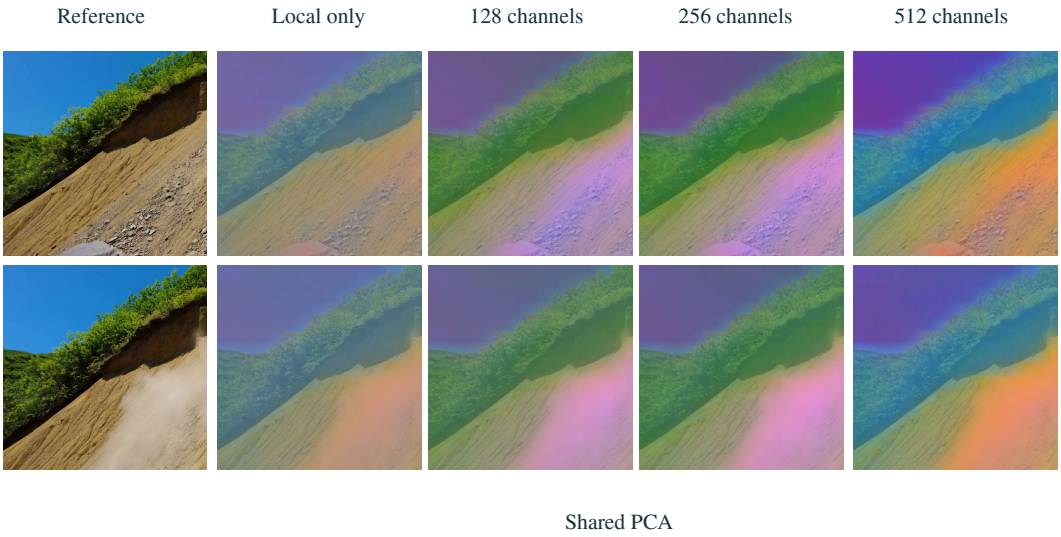


Figure 11: Transmission width changes the state available to an earlier cell. Two stages of a VideoPhy-2 rock descending a slope. Columns show the reference and the states h^i with local-only reading or 128, 256, and 512 transmitted channels, at a fixed x/r record and 512-channel local width. One PCA over the eight states fixes centering, component directions, and 99.5th-percentile absolute scales; RGB encodes three state coordinates over 28% reference texture. Figure 10 reports the corresponding completion.

Checking the counts against measurements. With the operating points of Table 7, the extra attention ratio Eq. (24) is 15.3% on Wan ($\rho = 0.083$, $N = 7$, $P_x = P_r = 5$), 7.2% on MAGI ($\rho = 0.044$, $N = 4$, $P_x = P_r = 65$), and 10.8% on Hunyuan ($\rho = 0.063$, $N = 5$, $P_x = P_r = 5$). The measured increases of final latency in Table 2 are 20.9%, 9.1%, and 17.8%, which rank the three backbones in the same order as the count. The measured values exceed the count by 1.9–7.0 points; the excess contains the backward scan and, on Hunyuan, the four-device partitioning, neither of which the attention count includes. The storage formula reproduces Table 2 exactly: for Wan, $2 \cdot 2 \cdot 30 \cdot 1536 \cdot 7 \cdot (4680 + 390)$ bytes are 6.54 GB, against 6.04 GB without revisions.

D Implementation: Algorithm, Architecture, and Training

This section describes how AmendAR is built on the three backbones and why each implementation choice is made. Appendix D.1 gives the complete algorithm, Appendix D.2 the revision layout and decoder, Appendix D.3 the targets and training stages, Appendix D.4 the adapters to each backbone's native sampler, and Appendix D.5–D.6 the controls, interventions, and external systems used in the experiments.

D.1 Complete Algorithm

Algorithm 1 lists generation, backward rendering, and both training stages in the notation of Section 4. $\text{KV}_\theta(\cdot)$ denotes the key–value states written by one clean forward pass at $t = 1$, Ψ is the native sampler update of each backbone (Appendix D.4), and $\text{sg}[\cdot]$ stops gradients. Part A differs from native generation only in lines 9–13, which sample and append the revision; the video steps and their cache reads are the native ones. Part B is the only computation that changes displayed frames, and it never touches $\mathcal{C}$. Part C trains the codec before the generator, because the generator's revision branch needs fixed targets $\tilde{r}$ to learn from.

Algorithm 1 AmendAR: append-only generation, backward rendering, and same-draft training

Require: prompt c; chunk count N; generator θ with velocity heads v^x_θ, v^r_θ
Require: decoder $\phi = (P_c, P_r, f_\phi, P_o)$ with state size $n_h \times d_h$; sampler update Ψ
Require: flow grids $0 = t_0 < \cdots < t_{K_x} = 1$ (video) and $0 = t'_0 < \cdots < t'_{K_r} = 1$ (revision)

Part A: generation with an append-only cache

procedure GENERATE($c; \theta$)
  $\mathcal{C} \leftarrow \emptyset$ ▷ KV cache; rows are only appended
  for $i = 1, \ldots, N$ **do**
    $x^i_{t_0} \sim \mathcal{N}(0, I)$
    for $j = 0, \ldots, K_x - 1$ **do** ▷ queries only for the rows of x^i
      $x^i_{t_{j+1}} \leftarrow \Psi\big(x^i_{t_j},\, v^x_\theta(x^i_{t_j}, t_j; \mathcal{C}, c),\, t_j, t_{j+1}\big)$
    end for
    $x^i \leftarrow x^i_{t_{K_x}}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \mathrm{KV}_\theta(x^i)$ ▷ preview may decode x^i here
    $r^i_{t'_0} \sim \mathcal{N}(0, I)$ ▷ independent noise stream
    for $j = 0, \ldots, K_r - 1$ **do** ▷ $\mathcal{C}$ now holds $\mathcal{H}^{<i}$ and x^i
      $r^i_{t'_{j+1}} \leftarrow \Psi\big(r^i_{t'_j},\, v^r_\theta(r^i_{t'_j}, t'_j; \mathcal{C}, c),\, t'_j, t'_{j+1}\big)$
    end for
    $r^i \leftarrow r^i_{t'_{K_r}}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \mathrm{KV}_\theta(r^i)$
  end for
  return $x^{1:N}, r^{1:N}$
end procedure

Part B: backward rendering of a prefix of length k

procedure RENDER($x^{1:k}, r^{1:k}; \phi$)
  $h^{k+1} \leftarrow 0 \in \mathbb{R}^{n_h \times d_h}$
  for $i = k, k-1, \ldots, 1$ **do**
    $q^i \leftarrow P_c(x^i)$; $u^i \leftarrow P_r(r^i)$; $h^i \leftarrow f_\phi\big(q^i, [u^i; h^{i+1}]\big)$; $\hat{x}^{i|k} \leftarrow x^i + P_o(h^i)$
  end for
  return $\hat{x}^{1|k}, \ldots, \hat{x}^{k|k}$ ▷ VAE decodes these latents
end procedure

Part C: training

Require: prompts $\mathcal{P}$; teacher G^+; encoder E_ψ; decoder initialization ϕ_0; steps $S_{\mathrm{I}}, S_{\mathrm{II}}$
$\mathcal{B} \leftarrow \big\{(c, x^{1:N}, y^{1:N})\big\}$, $x^{1:N} \leftarrow \mathrm{sg}[\mathrm{GENERATE}(c; \theta)]$, $y^{1:N} \leftarrow G^+(x^{1:N}, c)$ ▷ targets
for step $= 1, \ldots, S_{\mathrm{I}}$ **do** ▷ Stage I: codec
  draw $(c, x^{1:N}, y^{1:N}) \sim \mathcal{B}$ and $k \sim \mathcal{U}\{1, \ldots, N\}$
  $\tilde{r}^{\,i} \leftarrow E_\psi(x^{\le i}, y^{\le i})$ for $i \le k$; $\hat{x}^{1:k|k} \leftarrow \mathrm{RENDER}(x^{1:k}, \tilde{r}^{\,1:k}; \phi)$
  $(\psi, \phi) \leftarrow (\psi, \phi) - \eta_{\mathrm{I}} \nabla_{\psi,\phi} \frac{1}{k} \sum_{i=1}^{k} \|\hat{x}^{i|k} - y^i\|^2_{\mathrm{av}}$ ▷ Eq. (4)
end for
freeze ψ; $\phi \leftarrow \phi_0$; rerun lines 25–29 updating ϕ only ▷ decoder refit
for step $= 1, \ldots, S_{\mathrm{II}}$ **do** ▷ Stage II: generator
  draw $c \sim \mathcal{P}$; $(x^{1:N}, r^{1:N}) \leftarrow \mathrm{sg}[\mathrm{GENERATE}(c; \theta)]$ ▷ current-model draft
  $y^{1:N} \leftarrow G^+(x^{1:N}, c)$ ▷ enhance this exact draft
  $\tilde{r}^{\,i} \leftarrow E_\psi(x^{\le i}, y^{\le i})$; $\mathcal{H}^{<i} \leftarrow (x^1, r^1, \ldots, x^{i-1}, r^{i-1})$ for all i ▷ KV under current θ
  draw $t_x, t_r \sim \mathcal{U}[0,1]$ and $\epsilon, \epsilon' \sim \mathcal{N}(0, I)$ independently for each i
  $\mathcal{L} \leftarrow \sum_{i=1}^{N} \Big[\|v^x_\theta\big(t_x y^i + (1 - t_x)\epsilon,\, t_x; \mathcal{H}^{<i}, c\big) - (y^i - \epsilon)\|^2_{\mathrm{av}}$
    $+ \|v^r_\theta\big(t_r \tilde{r}^{\,i} + (1 - t_r)\epsilon',\, t_r; \mathcal{H}^{<i}, x^i, c\big) - (\tilde{r}^{\,i} - \epsilon')\|^2_{\mathrm{av}} \Big]$ ▷ Eq. (5)
  $\theta \leftarrow \theta - \eta_{\mathrm{II}} \nabla_\theta \mathcal{L}$
end for
return θ, ψ, ϕ

In our runs, $S_{\mathrm{I}} = 1{,}000$ with batch size 8, $|\mathcal{B}| = 4{,}000$ initial prompts, and $S_{\mathrm{II}} = 750$ with batch size 8 (6,000 current-model examples); AdamW replaces the plain gradient steps shown above (Appendix D.3).

D.2 Revision layout and decoder

Table 7 lists the native operating points. Wan uses the Self Forcing chunk-wise DMD EMA model built on Wan2.1 (Wan Team et al., 2025; Huang et al., 2025). MAGI uses 4.5B-distill with a serial single-chunk window and full clean history (Sand.ai et al., 2025). Hunyuan uses minWM's four-step causal conversion of HunyuanVideo 1.5 in text-only mode (Wu et al., 2025; Zhao et al., 2026). The longer settings use $N = 40$, 30, and 30 chunks, yielding 477, 720, and 477 frames at native frame rates. Computation uses BF16 with FP32 optimizer state. Wan and MAGI each run on one A100-80GB GPU; Hunyuan uses the same four-A100 partition for AmendAR and Native A-only.

Table 7: Native operating points. Wan/Self Forcing (Wan Team et al., 2025; Huang et al., 2025), MAGI-1 (Sand.ai et al., 2025), and HunyuanVideo 1.5/minWM (Wu et al., 2025; Zhao et al., 2026). Resolution is height × width; Chunks/lat. lists chunks × latent frames per chunk; A/R tokens are per-chunk video/revision counts n_x/n_r. KV layout is layers × key/value width. A/R steps are video/revision sampling steps, excluding clean passes. GPUs are A100-80GB; MAGI uses grouped-query attention.

Native backbone	Short output		Autoregressive layout				GPUs
	Resolution	Frames/fps	Chunks/lat.	A/R tokens	KV layout	A/R steps	
Wan2.1 / SF, 1.3B	480×832	81/16	7×3	4680/390	30×1536	4/4	1
MAGI-1, 4.5B-distill	720×720	96/24	4×6	12150/529	34×1024	64/64	1
HY1.5 / minWM, 8.3B	480×832	77/16	5×4	6240/390	54×2048	4/4	4

One revision site per 32×32-pixel region of a chunk. A revision describes how a finished chunk should change, and the events it has to fix, such as where a hand reaches or when an object falls, span many pixels and all frames of the chunk. We therefore place revisions on a grid that is coarser than the video tokens in space and has no time axis within a chunk. Each revision site covers 4×4 latent pixels on Wan and MAGI (VAE stride 8) and 2×2 latent pixels on Hunyuan (VAE stride 16), which is a 32×32-pixel region on all three backbones, over all latent frames of the chunk. The grids have $n_r = 15 \times 26 = 390$, $23 \times 23 = 529$ (the 90×90 MAGI latent is zero-padded to 92×92), and 390 sites. The content projection P_c reads the corresponding video patch of the whole chunk, giving input widths $3 \cdot 4 \cdot 4 \cdot 16 = 768$, $6 \cdot 4 \cdot 4 \cdot 16 = 1536$, and $4 \cdot 2 \cdot 2 \cdot 32 = 512$. Revisions have 64, 64, and 32 channels per site, the same width as one video token, so the revision of a chunk has exactly the fraction $\rho = n_r/n_x$ of the chunk's scalars: 8.3%, 4.4%, and 6.3%. This small size keeps the second sampler and the extra KV cheap, which is what the cost counts of Appendix C.6 use.

Decoder. The decoder state h^i lives on the same grid ($n_h = n_r$) with $d_h = 512$ channels, so every site of an earlier chunk receives a state vector from the same image region of the later chunk. The encoder E_ψ and decoder cell f_ϕ each use four Transformer blocks with hidden width 512, eight heads, feed-forward width 2048, GELU, and pre-normalization. Linear maps include biases, and positions use fixed sinusoidal encodings. Decoder blocks normalize query, key/value, and feed-forward inputs separately; encoder blocks use two affine normalizations, and both networks end with an affine normalization. P_o maps each site back to its video patch and removes padding before the native-coordinate MSE is computed. The cell has the same parameters at every chunk, so one decoder renders records of any length, including the 30–40 chunks of the long setting. On the generator side, the new parameters are two type embeddings that mark rows as video or revision and the revision input and output maps; all Transformer blocks are shared, so the revision branch reuses the backbone's text and time conditioning.

D.3 Targets and training stages

Prompts. We use VidProS, the 1,038,805-prompt semantically deduplicated text subset of VidProM (Wang & Yang, 2024). Source-disjoint splits contain 12,000 training, 512 validation, and 64 calibration prompts. Official evaluation prompts and identifiable same-source derivatives are excluded, and related rewrites stay in the same split, so evaluation prompts never appear in training in paraphrased form.

Same-draft targets. The residual $\hat{x}^{i|N} - x^i$ can only learn a correction if the target shows the same scene as the draft. VEnhancer v2 (He et al., 2024) takes a generated video and its text and returns a spatially and temporally enhanced version of the same video, so it provides such targets. We apply it to each exact current draft with noise augmentation 200, 15 enhancement steps, CFG 7.5, seed 666, and the original frame rate. Area interpolation restores the native spatial size before VAE encoding,

which aligns the spatial sizes of x^i and y^i at matched frame indices. Enhancements of evaluation drafts are used only in the same-draft evaluation.

Stage I: codec. For each backbone and seed, the first $4{,}000$ training prompts provide initial drafts and targets. Stage I updates E_ψ and D_ϕ jointly for $1{,}000$ steps at batch size 8 with uniformly sampled prefix lengths, so that the decoder learns to render with any amount of later context, as streaming previews require. We then freeze E_ψ, reset D_ϕ to its saved initialization, and refit it for $1{,}000$ steps on the frozen encoder's revisions. Freezing gives every revision target a fixed meaning before the generator learns to sample it. Refitting from the saved initialization makes the deployed decoder, the no-u decoder, and the decoders of the width study start from the same weights and see the same targets, so their differences come only from what they are allowed to read.

Stage II: generator. Stage II fixes both codec networks, draws $6{,}000$ current-rollout examples from the training split with replacement, and performs 750 generator updates at batch size 8. Rollouts are regenerated with the current weights, because the revision has to correct the errors the current generator makes; Table 12 measures what a fixed source costs. Stored latent histories are re-encoded under the current weights, so the generator trains on the same keys and values it reads at inference. Rollouts are sampled without gradients: the teacher is not differentiable, and the targets need only the draft, not its sampling path.

Optimization. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.01, gradient clipping at one, and 5% warmup followed by cosine decay. Nominal learning rates are 10^{-4} for the codec and 10^{-5} for the generator; each is selected from 0.5, 1, and 2 times the nominal value by held-out validation MSE. Checkpoints are selected by codec MSE or $\mathcal{L}_x + \mathcal{L}_r$, with the same procedure for all controls.

D.4 Adapters to the native samplers

The generator keeps each backbone's own flow convention and sampler, so that the video branch starts from the native model and the only change at inference is the added revision step. Eq. (5) is written in one continuous convention, and the following adapters map it to each backbone.

For Wan and Hunyuan, the native convention is $x_\sigma = (1-\sigma)z + \sigma\epsilon$ with velocity $v_{\text{native}} = \epsilon - z$. The velocity in Eq. (5) equals $-v_{\text{native}}$ at $\sigma = 1-t$, with native time $1000(1-t)$. Training samples continuous t without rounding to a timestep table. The nominal inference fractions $(1, .75, .5, .25)$ are shifted by $g(\sigma) = 5\sigma/(1+4\sigma)$ to $(1, 15/16, 5/6, 5/8)$, followed by clean $\sigma = 0$. Wan predicts $\hat{z} = x_\sigma - \sigma v_{\text{native}}$, re-noises it independently at the next fraction for the first three calls, and returns the final clean prediction. Hunyuan executes four Euler updates $x_{\sigma'} = x_\sigma + v_{\text{native}}(\sigma' - \sigma)$ with CFG 1 and zero context noise. MAGI's native flow already runs from noise to data as in Eq. (5). Each MAGI branch uses 64 Euler intervals with $t_j = (j/64)^2/[3 - 2(j/64)^2]$, $j = 0, \ldots, 64$; the fixed distillation condition 32 is kept in training, sampling, and clean passes; CFG is 1 and FP8 is disabled. The VAE latent scaling is applied only at the VAE boundary, not to revision noise or velocities.

Two rules are common to all backbones. Each branch has its own random stream, including Wan's intermediate re-noising draws, because Eq. (2) samples x^i and r^i with independent noise. The cache is written from the exact clean endpoint by a clean pass at $t = 1$, so that the stored keys and values belong to the latent that later chunks and the decoder read, which is the condition of Proposition 2.

D.5 Controls and fixed-record interventions

Each experiment in Section 5.3 removes one possible explanation of the gain, and the controls are built for that purpose.

Matched training controls. AmendAR trains toward enhanced targets, and training a video-only model toward the same targets could produce the whole gain. Equal-update A-only therefore gives the native model 750 video updates on $6{,}000$ target examples, the same video training AmendAR receives. AmendAR also spends extra time on the codec and the revision branch, so equal-learning-time A-only receives the same parameter-learning GPU time and the same target availability: with G the GPU time of AmendAR's codec fitting, decoder refitting, and generator learning, and g_x the time of one A-only update, it performs $\lfloor G/g_x \rfloor$ updates on $4{,}000$ initial and $6{,}000$ current-source

examples in a fixed 2:3 ratio, reusing targets as needed. Teacher enhancement and source collection are reported separately in Table 9.

Component controls. The no-u control refits D_ϕ with the same encoder and targets while setting $u^i = 0$, so each chunk is corrected only from video content. The no-r-read generator is trained with video queries blocked from revision keys from the first chunk onward, which removes the forward path. The fixed-source control keeps initial-generator drafts throughout adaptation with matched target examples and codec rules. Each trained generator is evaluated on its own rollouts.

Fixed-record interventions. Any retrained control also changes the drafts, so its effect mixes a different record with a different rendering. The interventions avoid this by keeping one AmendAR record $(x^{1:N}, r^{1:N})$ and its KV cache fixed and changing only what the decoder reads. For a late revision at position j, we keep the original h^{j+1}, set $u^j = 0$, recompute $h^j, h^{j-1}, \ldots$, discard the temporary output at j, and replace only the outputs $i < j$, keeping AmendAR's outputs for $i \geq j$. Any change in the task score then comes from the earlier chunks alone. The short-video test uses one of $j = 2, \ldots, 7$ for Wan, $2, \ldots, 4$ for MAGI, or $2, \ldots, 5$ for Hunyuan at a time, without accumulating removals. The long test uses $j = N$ and replaces only the output of chunk 1. The full VAE then decodes and the full video is evaluated, so temporal VAE receptive fields can affect neighboring frames. The no-incoming-state test sets $h^{i+1} = 0$ at every cell while keeping x^i and r^i, which removes both later video and later revision information.

Transmission width. To test the rank bound of Proposition 2, the width of the state has to change while everything else stays fixed. E_ψ, the generator, and the sampled record are frozen. Each decoder is refitted from the same saved initialization for 1,000 updates with its local hidden width and parameter count unchanged; only the state sent to the preceding cell is channel-masked to 128, 256, or 512 channels per site. Wan therefore transmits 49,920, 99,840, or 199,680 scalars per boundary, and MAGI 67,712, 135,424, or 270,848. Dense tensors stay allocated at every width, so the manipulated quantity is the transmitted size m alone. Encoded and sampled revisions are tested separately.

Figure 12 shows what one such intervention does to a video: the record and the cache are unchanged, and only the late revision input is removed from the decoder. The two renderings of each sequence differ little, in line with the 2–10% score changes of Figure 6, and the differences lie in the stages where the event progresses, such as how far the pink has spread from the petal bases of the sunflower.



Figure 12: Backward rendering on action sequences and color transitions. Upper pair: a refrigerator sequence; lower pair: a sunflower changing from yellow to pink. Within each pair, the full rendering (Full late read) and the rendering with the late revision input removed (Late input removed) are shown at eight corresponding stages. The protocol is the fixed-record intervention of Appendix D.5; aggregate results are in Figure 6.

All fixed-record interventions act on the decoder inputs of one unchanged record: the local revision u^i, the incoming state h^{i+1}, or a single late revision u^j. Because the generator and its samples stay

the same, each measured difference belongs to the backward path alone, which is why these tests complement the retrained controls above.

D.6 External systems

The external comparison is at the system level: each method keeps the training and history policy it was designed with, because these policies are what the methods contribute. Rolling Forcing uses five-step rolling sampling with a 15-latent-frame active window and $3+3$ global/recent clean frames (Liu et al., 2026). LongLive keeps three sink and nine local frames with its long-rollout training (Yang et al., 2026). Context Forcing keeps $3/12/6$ sink/slow/fast frames at similarity threshold 0.95 with its long-context teacher training (Chen et al., 2026). Deep Forcing uses deep sink 10, recent context 4, and the original 21-to-16 compression cycle, counting the actual peak before compression (Yi et al., 2026). Pathwise TTC keeps correction and re-noising at nominal steps 500 and 250, with every additional call timed (Xiang et al., 2026). Causal Forcing keeps its causal teacher, ODE initialization, and distillation recipe (Zhu et al., 2026).

Our Self-Forcing++ reproduction (Cui et al., 2026) uses EMA with the settings below. It starts from the public Self Forcing ODE initialization and filtered extended VidProM prompts, rolls out 201 latent frames, selects a 21-frame segment starting uniformly in $0, \ldots, 180$, and applies backward-noise-initialized DMD. Generator/critic learning rates are $2 \times 10^{-6}/4 \times 10^{-7}$, batch size is eight, and AdamW uses $(0, 0.999)$. Each outer step updates the critic, with a generator update every fifth step. EMA starts at step 200 with decay 0.99. The teacher and initial critic use the Wan 1.3B model with teacher CFG 3, and GRPO is disabled. Training ends at $15{,}000$ generator and $75{,}000$ critic updates with independently drawn batches, and evaluation uses the final EMA.

E Evaluation protocol

The evaluation has to answer two questions: whether requested events finish more often, and whether other properties of the video stay at the backbone's level. The benchmarks are chosen for these two roles (Appendix E.1). Every internal comparison is paired by seed, and the intervals reflect variation across independent training runs (Appendix E.2). Cost is measured as latency, KV memory, and adaptation GPU time (Appendix E.3).

E.1 Benchmarks and scoring

Table 8 lists the evaluation populations; every benchmark is used with its full prompt set.

Table 8: Evaluation populations per seed. Per text is the number of outputs. VBench has 946 category registrations but 944 unique texts. Repeated outputs share the seed's model.

Benchmark	Prompts	Per text	Videos
VBench (S/L)	944	5	4720
CompBench V2	1400	1	1400
VideoPhy-2	590	1	590
TC-Bench	150	5	750
StoryEval	423	1	423
ChronoMagic	1649	3	4947

Event completion. VideoPhy-2, TC-Bench, and StoryEval score the failure that motivates AmendAR: a video in which the early frames do not lead to the requested outcome. VideoPhy-2 uses all 590 test prompts unchanged. Its automatic evaluator reads 32 frames in BF16 with deterministic decoding, temperature 0.001, and maximum output length 256; physical scoring receives the video only. Semantic and physical scores lie on the native one-to-five scale, and joint success requires both to be at least four, so a video counts only if the action happens as described and looks physically plausible. Parser failures contribute the official zero outcome.

TC-Bench uses all 150 prompts with five outputs each. A blinded protocol assigns three ratings per video from an eight-person qualified pool, using Q1 for transition completion and Q2 for overall text alignment on one-to-five scales. Paired methods use the same rater triplet and independently randomized presentation order, so rater differences cancel within a pair. Completion requires a mean Q1 rating of at least $11/3$, equivalently a three-rating integer sum of at least 11. Threshold curves use all 13 attainable three-rater means $l/3$, $l = 3, \ldots, 15$, with survival probabilities $P(Q_1 \geq l/3)$ and $P(Q_2 \geq l/3)$; they show whether a gain occurs at one cutoff or across the rating range. StoryEval uses 423 prompts with 133 two-event, 262 three-event, and 28 four-event cases. LLaVA-OneVision-

Qwen2-72B reads 32 uniformly sampled frames with the official description-then-event procedure, an event succeeds only if both repeated judgments agree, and the score averages each prompt's fraction of completed events.

Relations, quality, and motion. T2V-CompBench V2 has seven categories of 200 prompts with one output per prompt. We report each category separately, without an overall score, because action and interaction binding relate to event completion while the other categories do not. Native scoring combines MLLM judgments (using category-specific settings) for attribute, action, and interaction binding with detection, tracking, or counting for the other categories.

VBench uses 944 unique prompts (its 946 metadata registrations contain two repeats, which reuse the same outputs) with five outputs each. Its sixteen dimensions are combined with the official normalization bounds without extra clipping: the first seven form quality, with half weight on dynamic degree, the remaining nine form semantic alignment, and the total is 0.8 quality $+\,0.2$ semantic. The $\sim$30 s setting uses the standard long-video mode with the mixed clip-length configuration, five samples per prompt, and no scene cuts. Subject and background consistency average the official slow and mapped fast branches, slice-based dimensions keep native slice-level aggregation, flickering keeps valid static slices under the native filter, and color keeps its detected-object denominator.

ChronoMagic uses 1,649 prompts with three outputs each. UMT-FVD is computed from all 1,649 reference and 4,947 generated feature vectors with the native 768-dimensional UMT representation and covariance normalization by the sample count. UMTScore, MTScore, and CHScore are averaged over all generated videos, and CHScore uses the paper's 0.1 scaling. MTScore and CHScore are read together, because a static video can be temporally consistent while showing no metamorphosis. All compared outputs are scored under the same protocol.

E.2 Independent units and reported uncertainty

The independent unit of an internal comparison is a training run. Seeds 17, 29, and 43 define three batches; for internal conditions each seed controls both training and generation, whereas a published checkpoint contributes generation variation only. Conditions are paired on the seed, evaluation example, output repetition, chunk, and sampling branch. Repeated outputs and repeated judge calls within one batch refine the score of that batch but do not add independent runs. For batch scores a_1, a_2, a_3, the reported SD is $s = \sqrt{\sum_j (a_j - \bar{a})^2/2}$. For a paired comparison with scores b_j, we compute $\delta_j = a_j - b_j$ from unrounded values and report the two-sided 95% Student-t interval with two degrees of freedom,

$$\bar{\delta} \pm 4.30265273\, s_\delta/\sqrt{3}. \tag{25}$$

With three runs the multiplier is large: Wan's VideoPhy-2 margin over the time-matched control has mean 2.7 points and a run-to-run SD of 1.2 points, and its interval, -0.3 to 5.6, still includes zero. These intervals describe variation across training runs under the pairing; they do not describe uncertainty over individual prompts or raters. Tables show three-batch means, figures show three-seed means, and the capacity figure also shows sample SDs. Threshold points reuse the same videos within each seed, and latent MSE uses the same draft and teacher target within each comparison.

E.3 Latency, memory, and adaptation cost

A streaming user sees the first chunk as soon as it is decoded, and a final user sees the complete video, so we time both. Timing uses warmed-up models at generation batch size one and starts with text processing. Preview latency ends after the first video chunk and one VAE preview, before AmendAR's first revision step; final-output latency includes all video and revision sampling, clean passes, the preview, the backward scan, and complete VAE decoding, and ends at the generated frames, before evaluation and disk encoding. On Hunyuan the timing includes the four-device partitioning overhead.

Table 9: Adaptation cost (GPU-hours). Three-batch means. Learning includes codec fitting/refitting and video/revision generator updates. Total covers these adaptation stages only, excluding pretraining and evaluation. Backbone specifications are in Table 7.

Backbone	Learn	Teacher	Rollouts	Total
Wan	14.1	474.0	34.1	522.2
MAGI	66.7	1063.8	2671.9	3802.4
Hunyuan	107.9	642.6	630.1	1380.6

Persistent visual KV is an analytic byte count from the retained tensor layouts: keys and values across all devices and owned copies, at $b = 2$ bytes per BF16 scalar. For a full history it equals $2bLd_{\mathrm{kv}}N(n_x + n_r)$; windowed or compressed methods use their retained row counts. Values are decimal GB and cover visual KV only. Adaptation cost separates target enhancement, source rollouts, parameter learning, and current-weight history re-encoding; GPU time sums over participating devices, so one hour on four GPUs counts as four GPU-hours, and shared stages are counted once.

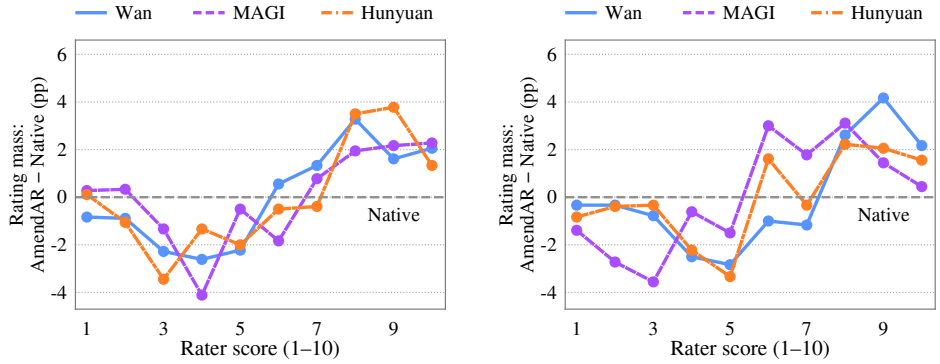
Most adaptation time builds targets. Table 9 separates parameter learning, teacher enhancement, and source rollouts. Parameter learning takes 14–108 GPU-hours, less than 10% of each total. Wan's teacher enhancement accounts for 474.0 of 522.2 GPU-hours, and MAGI's source rollouts for 2671.9 of 3802.4, because each MAGI rollout uses 64 sampling steps per branch. The adaptation cost of AmendAR is therefore set by the teacher and by the sampling cost of each backbone. The equal-learning-time control matches the parameter-learning component, and inference latency is measured separately (Appendix E.3).

F Results beyond the main text

The measurements that belong to one mechanism are reported next to its derivation: the backward path in Appendix A.4 and the decoder state in Appendix C.5. This section reports the remaining results in the order of the questions of Section 5: which videos improve and what else changes (Appendix F.1), how training and the forward path contribute to the gain (Appendix F.2), and what limits the correction (Appendix F.3).

F.1 Which videos improve, and what else changes

Mid-rated videos become high-rated. A higher completion rate can come from videos that fail entirely becoming correct, or from videos that almost succeed being pushed over the threshold. Revising earlier chunks should mainly produce the second kind, since it adjusts the setup of an event that the draft already attempts. Figure 13 shows the change of the rating distribution for T2V-CompBench action and interaction binding: with $p_{g,j}(l)$ the fraction of the 600 automated judgments of category g and seed j at level l, it plots the seed average of $100[p_{g,j}^{\mathrm{AmendAR}}(l) - p_{g,j}^{\mathrm{Native}}(l)]$. Across the six curves, levels 3–5 lose up to about 4 points of mass and levels 8–10 gain up to about 4 points. The two lowest levels change by about one point or less in five of the six curves; only MAGI interaction also loses mass there. Videos that the judge rates as partly correct thus move to high ratings, while videos with a wrong relation from the start mostly stay where they are, which matches a correction that repairs the setup of an attempted event.



(a) Action binding (b) Object interaction

Figure 13: Action and interaction gains redistribute rating mass. Each curve is the mean AmendAR-minus-Native difference in probability at the ten original T2V-CompBench V2 rating levels, in percentage points. Each category contributes 200 videos and three MLLM judgments per video per seed (600 ratings); three seeds are averaged. The plotted "rater" is the automated MLLM evaluator, separate from the human TC-Bench raters. The zero line denotes unchanged mass, and each curve sums to zero across rating levels.

Table 10: Complete complementary quality and motion comparison. Three-batch means for the same native and external systems as Table 1. VBench dimensions and CompBench consistent-attribute binding are on a 0–100 scale. ChronoMagic uses full-set UMT-FVD, native UMT/CH scaling, and MT in 10^{-2} units. Higher is better except FVD. Short and long clips are evaluated separately. Best means are marked within each backbone; † denotes independent reproduction. VBench total scores are in Table 1.

Method	VBench-short				VBench-long			Comp.	ChronoMagic			
	Subj.	Back.	Dyn.	Image	Subj.	Back.	Dyn.	Attr.	FVD↓	UMT	MT	CH
Wan2.1 / Self Forcing, 1.3B (Wan Team et al., 2025; Huang et al., 2025)												
Self Forcing (Huang et al., 2025)	96.20	96.94	49.17	69.19	91.92	94.25	37.81	64.29	161.2	3.231	41.7	43.49
Rolling Forcing (Liu et al., 2026)	95.71	96.61	52.13	68.81	92.66	94.44	53.93	63.70	156.8	3.244	**45.6**	42.74
Self-Forcing++† (Cui et al., 2026)	95.77	96.57	50.93	69.09	93.32	94.79	52.86	63.74	158.8	3.255	43.7	43.74
LongLive (Yang et al., 2026)	**96.81**	**97.29**	47.31	**70.16**	95.37	95.87	45.60	64.91	156.4	3.268	42.8	**44.51**
Causal Forcing (Zhu et al., 2026)	96.53	96.74	**52.96**	69.36	92.53	94.43	50.30	**65.65**	**156.0**	**3.280**	45.3	43.78
Context Forcing (Chen et al., 2026)	96.59	97.19	47.69	69.68	95.68	96.08	45.10	64.48	157.5	3.267	42.0	44.42
Deep Forcing (Yi et al., 2026)	96.36	96.95	48.98	69.28	**95.97**	**96.27**	57.04	64.14	159.9	3.240	43.3	43.45
Pathwise TTC (Xiang et al., 2026)	96.75	97.04	45.74	69.75	93.89	94.29	**59.11**	64.86	158.8	3.258	44.0	44.30
AmendAR (ours)	96.42	96.83	49.63	69.39	91.87	94.05	38.38	64.30	163.8	3.236	41.5	43.23
MAGI-1, 4.5B-distill (Sand.ai et al., 2025)												
Native A-only	94.91	95.46	**54.63**	70.81	91.25	92.82	46.94	62.50	155.4	**3.286**	**44.1**	41.67
AmendAR (ours)	**94.98**	**95.57**	**54.63**	**70.85**	**91.36**	**92.94**	**47.01**	**62.60**	**155.2**	3.284	42.8	**42.14**
HunyuanVideo 1.5 / minWM, 8.3B (Wu et al., 2025; Zhao et al., 2026)												
Native A-only	93.31	**94.22**	44.35	**68.36**	89.20	91.07	38.84	**58.52**	**180.5**	**3.015**	**35.8**	**39.18**
AmendAR (ours)	**93.55**	**94.22**	**45.09**	68.28	**89.24**	**91.13**	**39.00**	58.00	180.8	3.010	35.6	38.90

Complete systems. Figure 14 shows complete systems with and without AmendAR on Wan and Hunyuan for a forehand drive and for candles blown out by one breath. Each row is a separate trajectory, so scenes and subjects differ across rows, and the comparison concerns whether the requested event finishes.

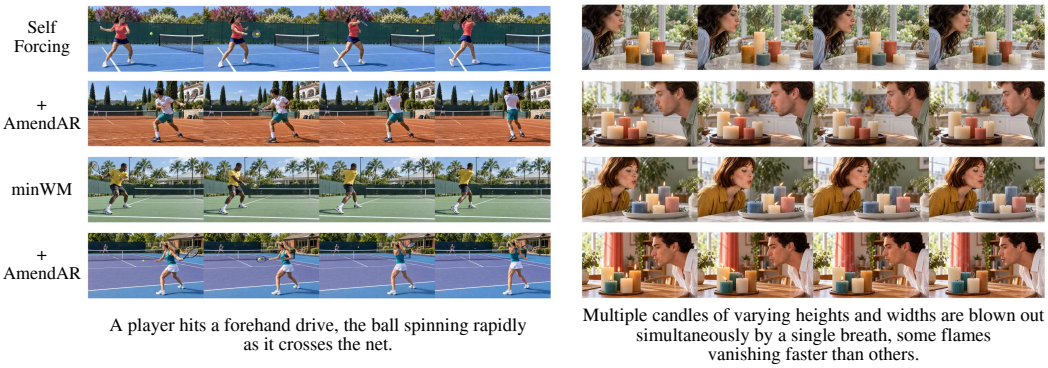


Figure 14: Independent full-system trajectories. Tennis (left) and candle extinction (right) are shown at four ordered stages. From top to bottom, rows correspond to Self Forcing, Self Forcing + AmendAR, minWM, and minWM + AmendAR. Each row is a separately generated trajectory; subjects and scene layouts vary across rows.

Other properties stay at the backbone's level. Table 10 reports VBench subdimensions, attribute binding, and all four ChronoMagic metrics for the systems of Table 1. Between each native model and AmendAR, VBench subdimensions change by at most 0.8 points, and most changes are below 0.3. The clearest regressions are Hunyuan's consistent-attribute score (58.5 → 58.0), Wan's UMT-FVD (161.2 → 163.8), and MAGI's MTScore (44.1 → 42.8). UMT-FVD measures the distance to ChronoMagic's reference videos, whereas AmendAR is trained toward VEnhancer versions of its own drafts, so the training target gives no pressure toward the reference distribution. The external systems show where AmendAR does not act. On VBench-Long, LongLive, Context Forcing, and Deep Forcing raise subject consistency to 95.4–96.0, against 91.9 for Self Forcing and AmendAR. These methods change which history later chunks see and how the generator is trained on long rollouts, while AmendAR keeps Self Forcing's history policy and changes how finished chunks are rendered, so long-video consistency and event completion improve under different methods.

F.2 WHERE THE GAIN COMES FROM

Section 5.3 attributes the gain to training toward enhanced targets, the forward path through the cache, and the backward path through the decoder. The backward path is measured in Appendix A.4; this subsection gives the evidence for the forward path and for training. The two are measured in different ways. The forward path is tested with the generator fixed, by blocking the revision keys of an otherwise unchanged model, so its effect is read from the prediction error; training is tested by retraining the native model under matched budgets, so its effect is read from the task scores.

The forward path matters most for multi-event prompts. Table 11 reports the fixed-query test of Section 5.3: video KV states, targets, noise, and flow times are shared within each pair, and only the revision keys are blocked. Blocking raises the flow-matching error in every cell. The increase is 3.5%, 4.5%, and 5.1% on VideoPhy-2 prompts for Wan, MAGI, and Hunyuan, and 5.1–7.1% on TC-Bench and StoryEval prompts. On every backbone the smallest increase occurs on VideoPhy-2, whose prompts mostly describe one action, and the larger increases occur on transitions and consecutive events, where a later chunk has to continue a corrected earlier chunk. By Eq. (18), these increases measure how much the trained predictor relies on the explicit revisions, beyond the revision information already contained in the video KV states.

Table 11: Fixed-query flow MSE↓ in 10^{-2} units. Three-batch means. Video KV, target, noise and flow time are identical within each pair. Only revision-history access changes; first-chunk queries remain included. Benchmark abbreviations follow Table 4.

Query context	VP-2	TC	StoryEval
Wan2.1 / Self Forcing, 1.3B			
Full revision read	**80.9**	**85.7**	**82.8**
Revision blocked	83.7	90.3	87.3
MAGI-1, 4.5B-distill			
Full revision read	**96.9**	**103.8**	**99.3**
Revision blocked	101.3	109.1	105.5
HunyuanVideo 1.5 / minWM, 8.3B			
Full revision read	**110.7**	**118.5**	**114.3**
Revision blocked	116.3	126.9	121.6

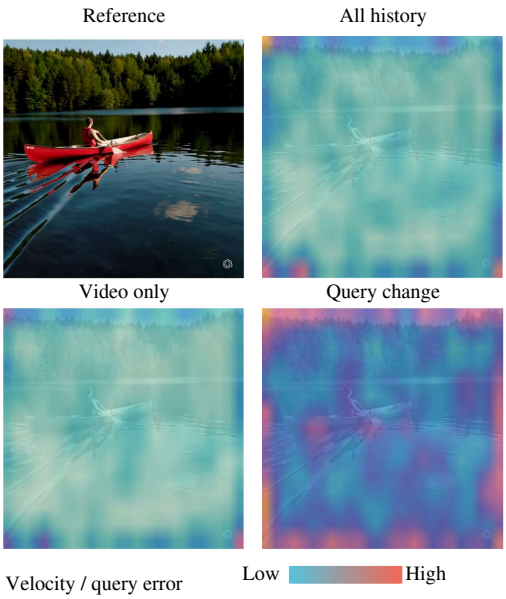
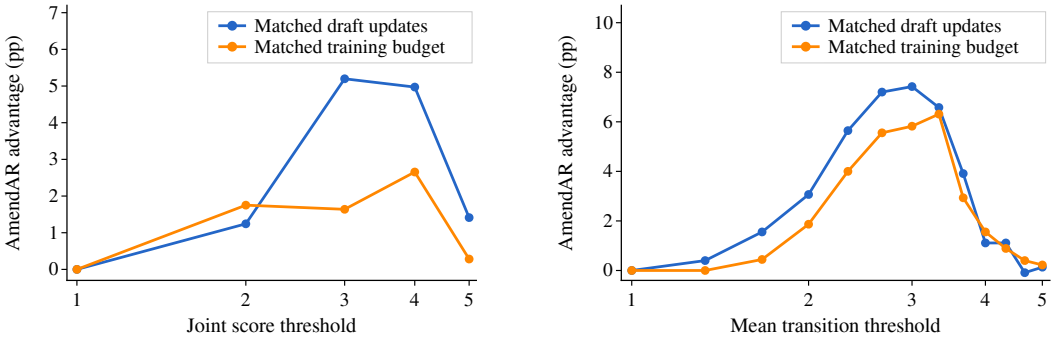


Figure 15: Fixed-query response to revision access. Reference, full-history error, video-only error, and query change share spatial coordinates.

Where a single query depends on the revisions. Figure 15 resolves one fixed query of this test spatially: a VideoPhy-2 canoe at chunk 4 and flow time $t = 0.44$. With target velocity $v^\star = z - \epsilon$ and d channels, it maps the errors $\|v - v^\star\|^2/d$ with full and video-only reading and the change $\|v_{\text{full}} - v_{\text{blocked}}\|/\sqrt{d}$ at each position. The change field shows where the prediction depends on revision access. In this query the full-reading error (0.04988) is higher than the video-only error (0.04184). A single query can go either way, because Eq. (18) contains fitting errors of both conditions; the aggregate in Table 11 averages over all queries, and there blocking raises the error.

Training on enhanced targets explains part of the gain. Figure 16 compares AmendAR with the two matched A-only controls at every success threshold. At the lowest threshold every video passes and at the highest almost none do, so all curves start and end near zero; the differences lie in between. On TC-Bench the margin over the time-matched control peaks at about 6 points near a mean transition rating of 3.3, next to the reported threshold $11/3$, and falls below one point above 4.5. On VideoPhy-2 the margin over the time-matched control stays between 1.6 and 2.7 points for thresholds 2–4. The margin therefore sits where videos are close to completing the transition, which again matches a correction that pushes near-successes over the threshold. The equal-update control trails AmendAR by more at the middle VideoPhy-2 thresholds, so extra learning time recovers part of the gain, and the remainder requires revisions.



(a) Wan: physical and semantic joint success

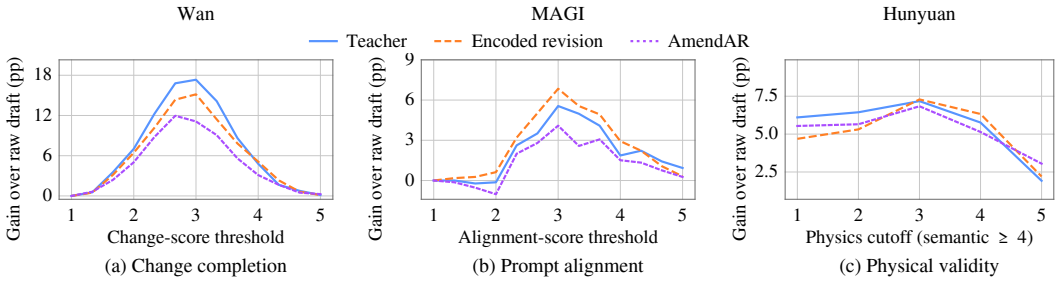
(b) Wan: human-rated transition success

Figure 16: Matched learning leaves threshold-dependent task gains. Wan AmendAR minus A-only controls, in percentage points, averaged over three paired seeds. Left: both VideoPhy-2 scores must reach the horizontal threshold (590 videos per seed). Right: the three-rater mean transition score must reach the threshold (750 TC-Bench videos per seed). The two controls match video updates or codec-plus-generator *parameter-learning* GPU time and target availability, respectively. Thus "matched training budget" refers to learning time; teacher enhancement and source collection are reported in Table 9. Lines connect attainable score thresholds.

F.3 What limits the correction

Eq. (11) names the possible limits: information the decoder cannot see, and error within the information it sees, which includes the transmitted state size measured in Appendix C.5. This subsection measures the sampler and the source of the drafts.

Sampling the revision loses more than encoding it. The same-draft comparison of Table 5 keeps the draft fixed and changes only the revision, so it separates the codec from the sampler. Figure 17 extends it over all thresholds. The ordinate is $100[P_a(\text{pass}\, l) - P_{\text{raw}}(\text{pass}\, l)]$ for output stage a and threshold l, computed within each seed before averaging; the first two panels use three-rater TC-Bench means (750 videos per stage and seed), and the third uses the 590 VideoPhy-2 videos with a fixed semantic criterion of at least four. All curves peak at middle thresholds, as in Figure 16. On Wan, the teacher gains about 17 points at the peak, decoded encoded revisions about 15, and sampled revisions about 12. On MAGI, decoded encoded revisions exceed the teacher itself for text alignment: the decoder output is its own estimate from the draft and the revision, which the judge can rate above the teacher's output. On Hunyuan, the three stages lie within about 1.5 points of each other for physical validity, and sampled revisions exceed both others at the strictest cutoff. The codec thus keeps most of the teacher's gain on every backbone, and the largest loss occurs when the generator has to sample the revision from its own history.



(a) Change completion

(b) Prompt alignment

(c) Physical validity

Figure 17: Teacher, encoded, and sampled corrections of the same drafts. Gains over the identical raw draft x for the teacher target y, decoded encoder revisions $\tilde{r}$, and decoded sampled revisions r (AmendAR), averaged over three seeds. Panels show Wan transition completion, MAGI text alignment, and Hunyuan physical validity jointly with semantic score ≥ 4. Human-rating thresholds use all 13 attainable means; physical cutoffs are the five native levels. The crossing teacher and encoded curves show that the codec's reconstruction target and the task rankings differ. Each panel compares stages within one backbone and task.

The three stages act on the same regions. Figure 18 shows where the three stages change the same draft. The encoded, sampled, and teacher change maps agree closely in location, so sampled revisions act on the same regions of the draft as the teacher; Table 5 shows that they nevertheless recover less of the teacher's task gain. This agreement follows from the residual form of Eq. (3): each stage adds a change to the same draft, and the draft determines where the event goes wrong, so the stages share where they act and differ in how well they correct these regions, which the task scores measure.

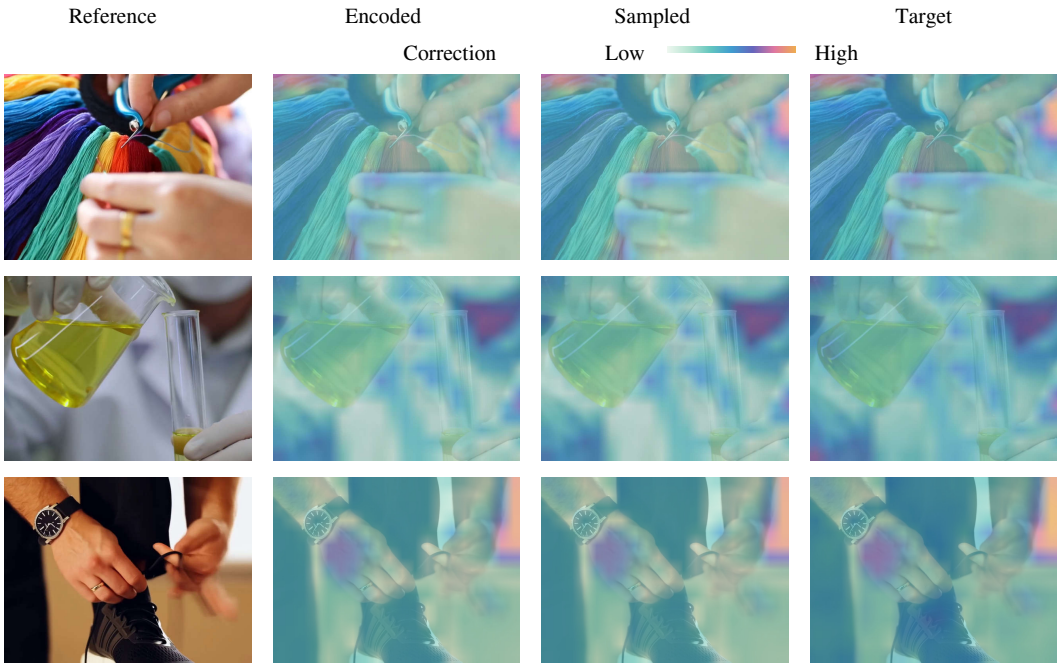


Figure 18: Where encoded and sampled revisions change the same draft. VideoPhy-2 needle threading, liquid pouring, and shoelace tying (rows). Columns: reference, decoding with encoded $\tilde{r}$, decoding with sampled r, and the teacher target y. Colors show each output's change from the same draft at spatial position p, $\|\hat{x}(p) - x(p)\|/\sqrt{d_x}$, with common divisor 0.3362816 across the nine fields. Each row shares its 4:3 region, and the 15×26 fields include 28% reference texture. Magnitude locates and sizes a change; task scores evaluate its effect.

Targets from the current model improve both drafts and correction. Training on drafts from the initial generator instead of the current one lowers rendered completion on every backbone and benchmark (Table 12). Part of this loss already appears in the raw drafts, 0.9–1.6 points. The rest is a smaller correction: the gain from rendering, rendered minus raw, is larger with current drafts by 0.8 and 0.9 VideoPhy-2 points and 1.3 and 0.7 TC-Bench points on Wan and MAGI. Hunyuan behaves differently on the two benchmarks: its rendering gain is 2.0 points larger with current drafts on VideoPhy-2 and equal on TC-Bench. Figure 19 resolves the TC-Bench contrast over thresholds. Wan and MAGI have positive contrasts at middle ratings, peaking at about 4–4.5 points near a rating of 3, whereas Hunyuan's contrast changes sign several times. At the reported threshold $11/3$, the contrasts are 1.38 ± 1.47, 0.71 ± 1.69, and 0.09 ± 0.89 points (mean $\pm$ SD).

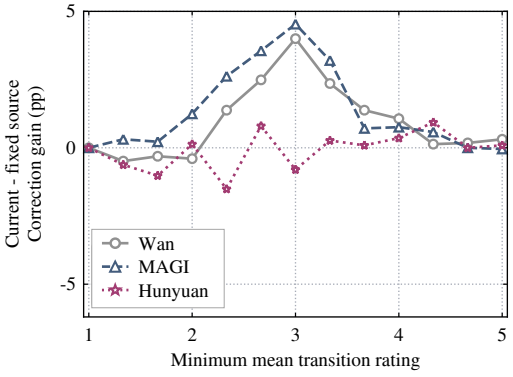


Figure 19: Source effects on correction depend on threshold. For TC-Bench, plot $(\text{current}_{\text{rendered}} - \text{current}_{\text{raw}}) - (\text{fixed}_{\text{rendered}} - \text{fixed}_{\text{raw}})$ in percentage points at each three-rater mean transition threshold. Each condition evaluates 750 videos per seed; contrasts are computed within seed and averaged over three seeds. Current and fixed-source generators produce their own drafts, so the curves describe incremental correction in their respective output distributions.

Table 12: Current versus fixed training sources. Completion percentages, three-batch means. Codec and target sampling are matched; each condition generates its own evaluation rollouts. Bold compares sources within the same readout, not raw versus rendered outputs. The three backbone configurations are given in Table 7.

Training source	VideoPhy-2↑			TC-Bench↑		
	Wan (1.3B)	MAGI (4.5B)	HY1.5 (8.3B)	Wan (1.3B)	MAGI (4.5B)	HY1.5 (8.3B)
Rendered outputs after correction						
Current-model drafts	**33.2**	**31.8**	**24.6**	**24.1**	**24.4**	**16.8**
Fixed initial drafts	30.9	29.4	21.7	21.5	22.1	15.4
Raw drafts before correction						
Current-model drafts	**27.6**	**27.6**	**19.5**	**18.5**	**20.1**	**13.5**
Fixed initial drafts	26.1	26.1	18.6	17.2	18.5	12.1

A current source helps because the revision is trained on the errors the deployed generator makes. Initial drafts contain errors that training has already removed and miss those it introduces, so a correction learned from them is aimed partly at errors the adapted generator no longer makes.

Sampled revisions share part of their direction across videos of one prompt. Figure 20 asks whether sampled revisions follow a consistent pattern for one prompt: for four independent inputs of TC-Bench prompt 100, a forest changing from green summer foliage to autumn colors, it compares the 64-channel revision vectors of all ordered input pairs at each normalized grid position p, $\kappa(p) = \frac{1}{12}\sum_{a\neq b}\cos\big(r_a(p), r_b(p)\big)$, and displays $(\kappa+1)/2$. Agreement varies across the frame, so part of the revision direction is shared by different videos of the same prompt, and part is specific to each draft. The grid positions correspond to image regions, which need not contain the same trees.

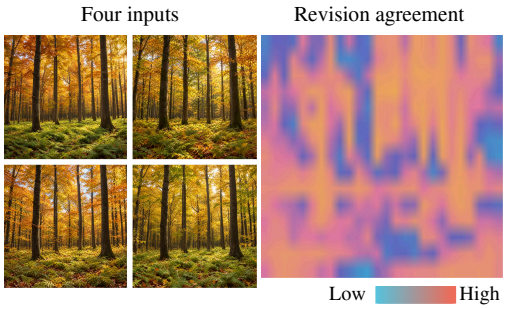


Figure 20: Four independent forest inputs and their shared revision field. Input pairs occupy the left column; the upper and lower halves of one agreement map occupy the right.

The same comparison on one video. Figure 21 renders one juggling draft with its encoded revision $\tilde{r}$ and its sampled revision r. The two renderings are close to each other and differ from the draft mainly in the positions of the hands and oranges in the second to fourth stages; this is the qualitative counterpart of the small gap between encoded and sampled revisions in Table 5.

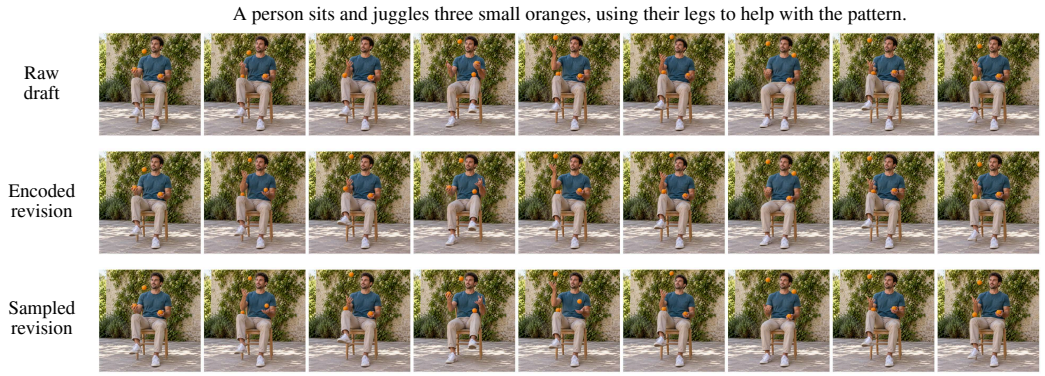


Figure 21: Encoded and sampled revisions on a shared draft. The prompt describes a seated person juggling three oranges with help from their legs. Rows show the raw draft x and its renderings with the encoded revision $\tilde{r}$ and the sampled revision r at nine corresponding stages. This fixed-draft comparison complements the aggregate stage comparison in Table 5 and Figure 17, and the spatial changes in Figure 18.