STREAMCEPTION: SELF-CORRECTING LATENT REPAIR FOR LONG STREAMING VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming video generation usually suffers from severe error accumulation when imperfectly generated latents are iteratively reused as fixed history. Inspired by variable-length generation paradigms like Flowception, we introduce StreamCeption, which adopts per-frame denoising and bidirectional attention but fundamentally shifts the generation paradigm. Instead of inserting latents into temporal gaps, StreamCeption consecutively appends noisy latents within a bounded active window, decoupling denoising completion from final frame output. While output frames become fixed history, the active window temporarily retains the oldest fully denoised latent. As subsequent trailing latents gain structural coherence during their denoising trajectories, a learned repair head adaptively determines whether to re-noise and refine this retained latent, correcting early errors before finalization. Trained alongside flow matching via self-sampled repair trials, our method uses reconstruction improvements to supervise repair selection and reconstruction loss to guide execution. Instantiated on the Wan2.2-TI2V-5B backbone, StreamCeption demonstrates the immense potential of this variable-length repair mechanism. Extensive experiments show that our approach scales effectively to this 5B model, achieving stronger long-horizon consistency than competitive streaming baselines while maintaining comparable visual quality, and seamlessly generalizing from an 81-frame training limit to highly coherent 162- and 320-frame videos with limited additional degradation over extended horizons.

1 INTRODUCTION

Video generation aims to synthesize visually realistic and temporally coherent videos from conditions such as text prompts and reference images. Recent advances in diffusion models have substantially improved visual fidelity and motion modeling (Ho et al., 2022; Esser et al., 2024; WAN et al., 2025). Within this field, *streaming video generation* extends videos incrementally in temporal order using a bounded temporal context (Huang et al., 2025; Kodaira et al., 2025), supporting applications such as interactive world simulation (PAN Team et al., 2025; AlayaWorld Team et al., 2026).

A common approach is *autoregressive diffusion*, new frames are conditioned on previously generated content. While forcing-based training Zhu et al. (2026a); Zhao et al. (2026); Liu et al. (2025); Zhu et al. (2026b); Cao et al. (2026) improves robustness, generated latents often retain artifacts. Once these imperfect latents become *fixed history*, they cannot incorporate information from subsequent frames, causing *error accumulation*. Notably, prior works tackle this via training strategies (Huang et al., 2025; Zhu et al., 2026a) and bidirectional denoising within a moving window (Liu et al., 2025; Kodaira et al., 2025). However, their samplers still freeze a latent once its denoising trajectory ends. Because they ignore later temporal context, residual errors in fully denoised latents still enter the fixed history and propagate to subsequent frames.

Our key insight is that *a fully denoised latent can still benefit from subsequent temporal context*. Early latents are finalized under an incomplete estimate of future frames. As later latents take shape, they offer complementary constraints to correct earlier inconsistencies. Yet, since later predictions may also contain errors, blindly refining early latents risks perturbing accurate content. This motivates learning *when* to revisit a fully denoised latent.

To address this, we introduce **StreamCeption**, which builds upon Flowception's (Berrada Ifriqi et al., 2025) variable-length bidirectional generation to selectively repair latents using subsequently

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.

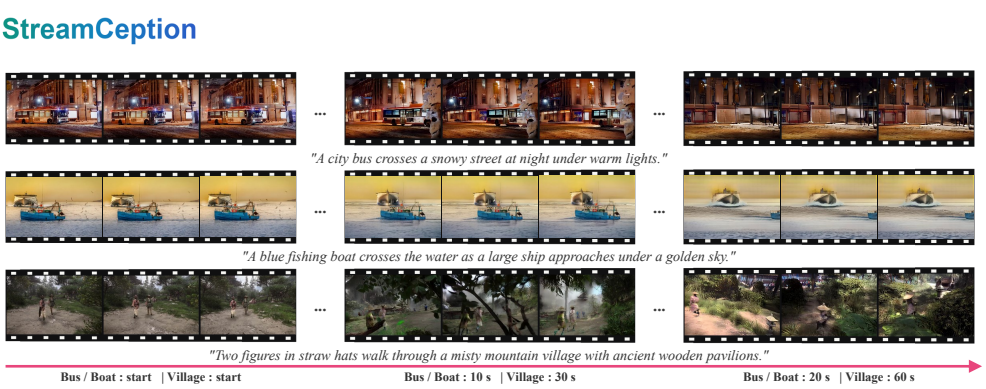


Figure 1: **Streaming video generation.** StreamCeption incrementally generates video via adaptive latent re-noising in a bounded window. Top rows (Bus/Boat) show 20s continuous outputs at 24 FPS; the bottom (Village) demonstrates a 1-minute output, verifying our stability over long horizons.

developed temporal context. We adapt this formulation by employing a bounded denoising window comprising immutable *fixed history* and unfinished *active latents*. Crucially, StreamCeption separates *denoising completion* from *final video output*. Once the oldest active latent becomes fully denoised, it is temporarily retained while newer latents undergo further denoising updates. A learned repair head then evaluates this retained latent against the newly enriched context and decides whether to keep it unchanged or re-noise it for further refinement via bidirectional attention. Finally, the latent enters the output sequence and becomes fixed history as the window advances.

During training, the model reconstructs partially noised ground-truth latents, keeping each reconstruction frozen while subsequent latents continue denoising. A sampled repair trial compares the unchanged and repaired versions of a reconstruction against its clean target. The measured improvement directly supervises the keep-or-repair decision, while the trial simultaneously optimizes the repaired latent toward the clean target. Combined with flow matching, these objectives jointly train repair selection and execution. During inference, the repair head relies solely on the available latent context and operates without clean targets or reconstruction-error computations.

We evaluate **StreamCeption** on text-and-image-conditioned streaming generation using VBench quality and image-conditioning metrics (Huang et al., 2024). We train the model on 81-frame clips at 24 frames per second (fps), lasting approximately 3.4 seconds. We evaluate generation at 5-, 10-, and 20-second intervals; the longest duration is roughly six times that of the training clips. At 5 seconds, **StreamCeption** achieves higher scores in subject consistency, background consistency, and motion smoothness compared to the Wan2.2-TI2V-5B and Rolling Forcing baselines, while maintaining strong input-image alignment. These advantages primarily lie in the consistency metrics, whereas Wan2.2 and Rolling Forcing retain the highest scores in aesthetic and imaging quality, respectively. Notably, **StreamCeption**'s performance metrics remain largely unchanged between 10 and 20 seconds, demonstrating robust generation capabilities well beyond the training horizon. Finally, qualitative ablations of the repair mechanism highlight representative cases where these corrections successfully preserve subject continuity as generation unfolds.

The main contributions of our proposed method are summarized as follows:

- We propose **StreamCeption**, a streaming video generation framework that separates denoising completion from final video output. By allowing fully denoised latents to incorporate subsequently developed temporal context within a bounded window, it effectively mitigates error accumulation in long-form generation.
- We introduce a selective repair mechanism that jointly learns repair selection and execution. Shared trials directly supervise both the keep-or-repair decision and the correction itself.
- We demonstrate robust streaming generation at roughly six times the training-clip duration. **StreamCeption** outperforms competitive baselines in benchmark metrics, maintaining stable generation quality without degradation even as sequences extend to 1 minute, see Fig. 1.

2 PRELIMINARIES

We review Flowception's variable-length generation and per-frame denoising formulation, which provides the foundation for StreamCeption's sequential expansion.

Flowception (Berrada Ifriqi et al., 2025) extends flow matching (Lipman et al., 2022) to growing sequences of latent frames by interleaving frame insertion with denoising. Its denoising model learns a velocity field conditioned on $\mathbf{c}$ by regression along a linear noise-to-data path. Flowception uses $t = 0$ for noise and $t = 1$ for the denoised endpoint. StreamCeption uses the equivalent noise coordinate $\sigma = 1 - t$ to express per-latent noise levels and re-noising strength. We denote velocities with respect to t by $\mathbf{v}$ and those with respect to σ by $\mathbf{u}$. For a clean latent $\mathbf{x}^\star$ and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the path and target velocity are

$$\begin{aligned} \mathbf{x}_\sigma &= (1-\sigma)\mathbf{x}^\star + \sigma\boldsymbol{\epsilon}, \\ \mathbf{u}^\star_\sigma &= \boldsymbol{\epsilon} - \mathbf{x}^\star = -\mathbf{v}^\star_t. \end{aligned} \tag{1}$$

Sampling integrates the learned velocity $\mathbf{u}_\theta$ as σ decreases from 1 to 0.

Flowception couples these continuous updates with discrete frame insertion, allowing the sequence length to increase during sampling. The content state of a nonempty sequence belongs to the *variable-length* space

$$\mathcal{X} = \bigcup_{m=1}^{\infty} \mathbb{R}^{m \times d}, \tag{2}$$

where m is the current frame count and d is the dimensionality of one latent frame. Sampling starts from a short sequence of Gaussian latents; $m = 1$ is the special case of a single noisy frame. New frames enter as noise, while existing frames retain their states and denoising progress.

Adding frames during denoising. Let $X = (X^1, \ldots, X^m)$ be the current sequence, with $X^i \in \mathbb{R}^d$, and let $\mathbf{t} = (t_1, \ldots, t_m)$ collect its per-frame denoising times. Inserting a fresh noise sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ after position i gives

$$\begin{aligned} X^+ &= (X^1, \ldots, X^i, \boldsymbol{\epsilon}, X^{i+1}, \ldots, X^m), \\ \mathbf{t}^+ &= (t_1, \ldots, t_i, 0, t_{i+1}, \ldots, t_m). \end{aligned} \tag{3}$$

The frame count increases from m to $m + 1$. The inserted frame starts at $t = 0$, while existing frames retain their states and denoising times. For each visible frame, the model predicts

$$\left(\mathbf{v}^\theta_i(X, \mathbf{t}, \mathbf{c}), \lambda^\theta_i(X, \mathbf{t}, \mathbf{c})\right). \tag{4}$$

The velocity $\mathbf{v}^\theta_i \in \mathbb{R}^d$ updates frame content, while the non-negative scalar λ^θ_i estimates the number of missing frames immediately after position i. A prescribed insertion schedule $\kappa(t_g) \in [0, 1]$ increases with global generation time t_g. For each frame outside the initial sequence, $\kappa(t_g)$ specifies the probability of having been inserted by time t_g. Flowception combines this schedule with λ^θ_i to determine the probability of adding a frame after position i. A Poisson count objective trains this scalar output using the number of absent frames in each gap, while flow matching supervises the velocities of visible, unfinished frames.

Here, *global generation time* t_g controls sequence expansion, whereas *local denoising time* t_i records the denoising progress of frame i.

Per-frame times and shared context. Insertion initializes only the new frame's time; existing frames retain their own denoising progress. Consequently, the current sequence can contain different noise levels. For a step size $h > 0$, let $\Delta_i = \min\{h, 1 - t_i\}$. A per-frame Euler update is

$$\begin{aligned} X^i &\leftarrow X^i + \Delta_i \mathbf{v}^\theta_i(X, \mathbf{t}, \mathbf{c}), \\ t_i &\leftarrow t_i + \Delta_i. \end{aligned} \tag{5}$$

All velocities are evaluated on the same current sequence, and bidirectional attention enables information exchange across frames at different denoising stages. Training reproduces these asynchronous states by assigning insertion delays to ground-truth frames. Frames whose insertion times have not yet arrived are *absent* from X. Among visible frames, $t_i = 0$ denotes newly inserted noise, $0 < t_i < 1$ denotes a partially denoised frame, and $t_i = 1$ denotes a completed frame. At the denoised endpoint, $\Delta_i = 0$: the original sampler no longer updates that frame, even as additional frames are inserted, and their temporal context develops.

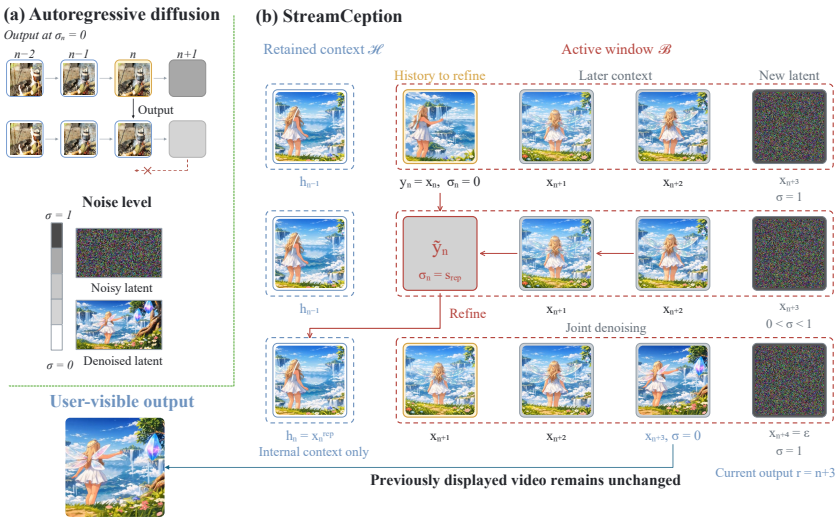


Figure 2: **StreamCeption overview.** (a) Autoregressive diffusion reuses completed predictions as fixed history, allowing their errors to affect subsequent generation. (b) StreamCeption appends noisy latents within a bounded window. Later context guides selective repair of an earlier internal latent, and the repaired context supports subsequent denoising. In this example, $\mathbf{x}_n$ is repaired internally while $\mathbf{x}_r$, with $r = n+3$, develops into the next video output. Only new completed output latents are decoded for display; previously displayed video remains unchanged. The rows summarize multiple updates and a window advance.

From Flowception to StreamCeption. While StreamCeption adopts Flowception's per-frame denoising and bidirectional attention, it fundamentally restructures the sequence expansion mechanism. Instead of learning to insert frames into arbitrary temporal gaps, thereby discarding Flowception's insertion prediction λ_i^θ, StreamCeption consecutively appends new latents within a bounded temporal window (as defined in Sec. 3.1). Within this window, sequence elements are strictly partitioned into two states: *fixed history* (previously output latents used solely as conditioning) and *active* (latents still undergoing flow updates). Crucially, when an active latent reaches $\sigma_i = 0$, its primary denoising trajectory concludes, but it is not immediately output. Instead, it remains active and available for further refinement. The following section details how StreamCeption selectively repairs this oldest active latent to ensure temporal consistency before finalizing its output.

3 Methodology

Streaming generation repeatedly conditions new predictions on earlier ones, so errors in the conditioning history can affect long continuations. A completed denoising trajectory need not produce a latent consistent with the developing sequence: later predictions can reveal motion and appearance information that was unavailable when an earlier latent finished denoising. This context consists of the model's own predictions. As illustrated in Fig. 2, StreamCeption uses this subsequently developed context to selectively refine internal history. The corrected context then guides further generation, while video already delivered to the user remains unchanged. This separates two requirements of streaming generation: temporal order constrains what can be displayed, while the model can still revise the internal context used to predict new content. Our design uses this flexibility to address error propagation during continuation.

3.1 Streaming Generation with a Bounded Context

Given an initial image I_1 and text prompt $\mathbf{c}$, we generate an ordered sequence of VAE latents $\widehat{\mathbf{x}}_{1:T}$, with $\widehat{\mathbf{x}}_1 = \mathbf{r} = \mathrm{Enc}(I_1)$. The target length T includes the image-condition latent. Starting with fresh noise, the sampler sequentially appends latents at consecutive temporal positions and jointly denoises them using bidirectional attention. Each latent follows its own noise level σ_i, where 1

denotes noise and 0 a denoised endpoint. New latents can enter before earlier ones finish, continually extending the sequence beyond a single model window.

The individual noise schedules allow an earlier latent to provide clean context while newer positions still develop their content. Ordinary denoising updates only unfinished latents. A completed latent remains available to the model, and changes again only if the repair mechanism selects it for refinement. These overlapping trajectories create the opportunity for correction: a later latent can acquire useful structure while an earlier prediction is still available in the same window. A single forward can then relate their motion and appearance through bidirectional attention, without requiring the entire requested video in memory.

As shown in Fig. 3(b), the model maintains retained context $\mathcal{H}$ and active working latents $\mathcal{B}$ within a bounded window:

$$\begin{aligned} &\mathcal{H} = [(\mathbf{h}_i, 0)]_{i=\max(1,n-P)}^{n-1}, \qquad \mathcal{B} = [(\mathbf{x}_i, \sigma_i)]_{i=n}^{m}, \\ &\mathcal{W} = \mathcal{H} \| \mathcal{B}, \qquad |\mathcal{H}| \le P < W, \quad |\mathcal{W}| \le W. \end{aligned} \tag{6}$$

Here n is the oldest active position eligible for repair and m the latest initialized position. Each $\mathbf{h}_i$ is the accepted internal latent retained after position i leaves $\mathcal{B}$; it is a context latent, not a transformer hidden feature. These values remain fixed while retained in $\mathcal{H}$. Active working latents can continue denoising or undergo selected repair. The initial-image reference $\mathbf{r}$ also conditions every window. The transformer input is $\mathcal{C} = (\mathbf{r}, \mathcal{H}, \mathcal{B}, \mathbf{c})$. Older retained values leave as the window advances, while the image reference remains available, keeping the model context bounded.

Internal context and video output. We distinguish the working latent $\mathbf{x}_i$ from its immutable output snapshot $\widehat{\mathbf{x}}_i$. Let r denote the next output position. The sampler maintains $n \le r$: repair targets position n, while new output is taken only from position r. To produce new video, the sampler copies the completed current latent at $\sigma_r = 0$ into $\widehat{\mathbf{x}}_r$ and appends this snapshot to the output sequence for VAE decoding. Output positions advance in temporal order; one latent may represent several RGB frames. For $n < r$, repair acts on the internal working copy of an already output position. Updating this copy changes the conditioning for subsequent generation, without replacing its output snapshot or previously displayed video. Its eventual retained value $\mathbf{h}_n$ may therefore differ from the output snapshot $\widehat{\mathbf{x}}_n$. Our streaming experiments include this $n < r$ setting. Figure 3(b) illustrates the distinction: internal repair at position n supports generation at the current output position $r = n + 3$. This offset is illustrative, rather than a fixed requirement. As internal candidates leave the active window, their accepted values become retained context and fresh noise enters. The context size remains bounded as the requested video length grows.

3.2 Selective Repair with Later Context

Developing context and selecting repair. As shown in Fig. 3(a), the *Hold* stage begins when the internal candidate at position n reaches $\sigma_n = 0$. We hold its value $\mathbf{y}_n = \mathbf{x}_n$ while later unfinished latents undergo a bounded number of additional denoising updates. Holding the candidate fixed lets its context develop before we reconsider its value: a newly appended noise latent initially provides little motion or appearance structure, but its later denoising predictions can supply complementary evidence. The relevant change is therefore the information available to the repair decision, even though the candidate itself has already reached zero noise. Later predictions can themselves be imperfect, and restarting denoising can disturb an adequate prediction. StreamCeption therefore learns whether a correction is beneficial under the available context. In the *Review* stage, a fresh decision forward evaluates the held candidate under the developed context. A binary head reads the candidate's features and predicts a keep-or-repair score ρ_n. Specifically, the head pools the candidate's spatial features, which incorporate neighboring latents through bidirectional attention, and predicts class probabilities $\mathbf{p}_n$. We use $\rho_n = p_{n,1}$ for the repair class. Its supervision measures whether a repair trial improves reconstruction, rather than whether the candidate has already reached zero noise. The head selects whether to repair; the re-noising strength is fixed. With repair enabled, an attempt requires $\sigma_n = 0$, $\rho_n > \tau_n$, and $a_n < A_{\max}$, where a_n counts completed attempts. The threshold is $\tau_n = \tau$ before any repair and $\tau_n = \tau_{\text{acc}}$ thereafter. Our reference configuration uses $\tau = 0.5$, $\tau_{\text{acc}} = 1$, and $A_{\max} = 1$, so a triggered repair is performed once and its completed correction is retained.

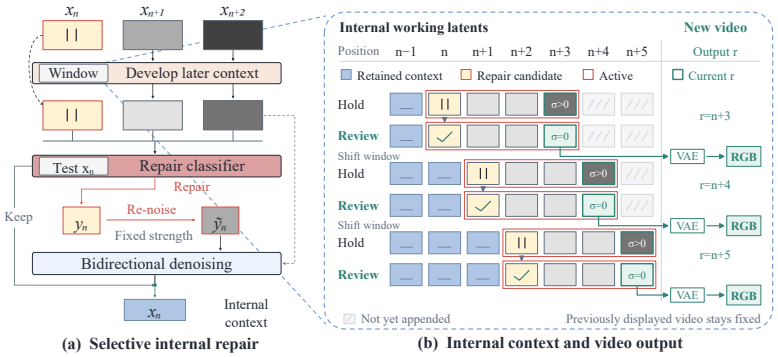


Figure 3: **StreamCeption inference.** (a) The fully denoised internal candidate $\mathbf{x}_n$ is held while later context develops. A binary head selects keep or repair; repair applies fixed-strength re-noising followed by bidirectional refinement. Both branches supply internal context. (b) The active window (red outline) advances while internal repair and video output target separate positions in this example. Blue cells denote retained context, yellow cells mark the repair candidate, and teal outlines identify the current output position r. Hatched cells have not yet been initialized. Only the completed latent at r is copied into the output sequence and decoded into new RGB frames. Repairing an earlier internal copy leaves previously displayed video unchanged. Each pair of *Hold* and *Review* rows summarizes multiple model updates; the initial offset $r = n + 3$ is illustrative.

Repairing internal history. For a selected candidate, we restart denoising at a small positive noise level s_{rep}, obtained by mapping the nominal strength 0.10 to the nearest positive inference-grid level. Re-noising allows the denoiser to revise the candidate under the developed context. Each repair uses fresh Gaussian noise:

$$\widetilde{\mathbf{y}}_n = (1 - s_{\text{rep}})\mathbf{y}_n + s_{\text{rep}}\boldsymbol{\xi}_n, \qquad \boldsymbol{\xi}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{7}$$

Bidirectional attention allows later context to guide a conditional prediction of the repaired endpoint:

$$\mathbf{x}_n^{\text{rep}} = \widetilde{\mathbf{y}}_n - s_{\text{rep}}\mathbf{u}_{\theta,n}(\widetilde{\mathcal{C}}), \tag{8}$$

where $\widetilde{\mathcal{C}}$ contains the re-noised candidate and updated neighboring latents. We replace the working value by $\mathbf{x}_n^{\text{rep}}$ at $\sigma_n = 0$. Other completed latents provide clean context, while unfinished neighbors continue denoising. The updated working value conditions subsequent generation; if $n < r$, its existing output snapshot $\widehat{\mathbf{x}}_n$ remains unchanged and is not decoded again. The general controller retains the current internal value when the score meets the applicable acceptance criterion. Only when further repair is requested after exhausting $A_{\max}$ do we restore the evaluated active-window snapshot with the lowest repair score, including the original candidate among the choices. This is a stopping heuristic based on remaining repair preference, not a ranking of reconstruction accuracy. It trades further computation against retaining an evaluated state. Always reverting to the original candidate would discard every correction at budget exhaustion, while always keeping the last attempt would ignore earlier evaluations. Reusing the existing scores provides a bounded stopping rule without introducing another quality evaluator; it does not guarantee the most accurate candidate. Appendix A.2 specifies the update ordering and restoration details.

3.3 Learning Repair Selection and Execution

Training combines flow matching with repair trials on the model's own reconstructions. For a clean training latent $\mathbf{x}_j^\star$, we partially noise it and use the current model to obtain a detached reconstruction $\mathbf{y}_j$. We retain the original clean latent as its target, hold the reconstruction fixed, and further denoise later latents to develop the context used for the repair decision. This constructs examples of model errors with known targets, approximating the internal states encountered during streaming inference. Using a model reconstruction exposes the repair model to errors left by its own denoising process. Keeping the source latent as the target makes the benefit of a trial directly measurable, while developing later context before that trial reproduces the information available at inference.

Every selected training probe receives a repair trial regardless of the head's current decision, providing supervision for both beneficial and harmful corrections. The head predicts probabilities $\mathbf{p}_j$ to

select repair, while endpoint supervision teaches the denoiser how to perform the correction using the same trial.

A sampled repair trial applies Eqs. 7–8 to obtain $\mathbf{x}_j^{\text{rep}}$. Its reconstruction improvement provides supervision for the decision:

$$R_j = \text{MSE}(\mathbf{y}_j, \mathbf{x}_j^\star) - \text{MSE}(\mathbf{x}_j^{\text{rep}}, \mathbf{x}_j^\star), \qquad b_j = \mathbf{1}[R_j > 0]. \tag{9}$$

The label asks whether this repair helps the given candidate, instead of whether the candidate is accurate in isolation. A flawed prediction need not benefit from a particular repair. The magnitude $|R_j|$ gives larger improvements and larger degradations more weight, so the head learns an improvement-weighted preference for repair. Weights are raw and unclipped; the loss averages over the number of trials, without normalization by the sum of weights. The same trial trains the repaired endpoint toward the original target, including when that trial does not improve reconstruction. The two objectives address different needs: classification learns to avoid an unhelpful action, while reconstruction learns to execute the correction more effectively. A harmful trial therefore supplies a keep label and still contributes to learning recovery toward the clean target. Over sampled trials $\mathcal{P}$, the repair objectives are

$$\mathcal{L}_{\text{cls}} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} |R_j| \, \text{CE}(\mathbf{p}_j, b_j), \tag{10}$$

$$\mathcal{L}_{\text{rep}} = \frac{1}{|\mathcal{P}| s_{\text{rep}}^2} \sum_{j \in \mathcal{P}} \text{MSE}(\mathbf{x}_j^{\text{rep}}, \mathbf{x}_j^\star). \tag{11}$$

Together with native flow matching, they give

$$\mathcal{L} = \mathcal{L}_{\text{FM}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{rep}} \mathcal{L}_{\text{rep}}. \tag{12}$$

Flow-matching supervision remains on the original ground-truth-noised states. Repair inputs additionally contain the candidate's existing reconstruction error, which is why the repair endpoint receives direct supervision rather than the original Gaussian-path velocity target. The s_{rep}^{-2} normalization expresses this endpoint loss as an equivalent correction-velocity loss, as derived in the appendix. Reconstruction collection and context development use detached predictions, and rewards provide detached decision targets. At inference, repair selection uses only the available latent context, without clean targets or reconstruction-error computation. Appendix A.3 provides the state construction, gradient paths, and loss normalization details.

4 Experiments

Experimental setup. We study text-and-image-conditioned streaming video generation, motivated by visual world simulation that continues a scene from an initial observation. The input image specifies the initial subject appearance and scene layout, allowing us to assess whether these properties remain consistent as generation continues. We compare StreamCeption with Rolling Forcing (Liu et al., 2025), Causal Forcing (Zhu et al., 2026a), and Mask Forcing (Zhao et al., 2026). These baselines cover rolling-window joint denoising and autoregressive diffusion distillation, broadening the comparison across approaches to streaming generation. Wan2.2-TI2V-5B (WAN et al., 2025) additionally serves as a short-video reference for the underlying backbone. We adapt the evaluated Mask Forcing checkpoint to accept image conditioning because it does not natively support the I2V setting. Table annotations identify this adaptation and the training configurations used for comparison.

For training, we construct approximately 300,000 video clips from `OGameData` (Che et al., 2025) and additional collected videos. Each clip contains 81 frames at 24 frames per second, or approximately 3.4 seconds, with the first frame serving as the image condition. We initialize StreamCeption from Wan2.2-TI2V-5B and fine-tune it using low-rank adaptation (LoRA) with rank 64, a learning rate of 1×10^{-5}, and a 100-step warmup. Training combines native flow matching with the objectives for repair selection and execution. At inference, StreamCeption extends the sequence one latent frame at a time within a bounded denoising window. We evaluate generation at 5, 10, and 20 seconds to examine performance beyond the training-clip duration.

Table 1: **Selected VBench-I2V results.** * denotes prompt-based supervised fine-tuning. † denotes configurations using a 14B teacher model as the prior. ‡ denotes an I2V adaptation of a checkpoint without native I2V support. Scores are multiplied by 100, and higher is better. Bold marks the highest displayed value in each column, including ties.

Method	Duration	I2V-S	I2V-B	SC	BC	MS	AQ	IQ
Wan2.2-TI2V-5B (I2V)*	5 s	**99.14**	**99.58**	94.95	96.81	98.65	**64.01**	70.70
Rolling Forcing†	5 s	98.69	98.96	95.03	95.42	98.78	62.94	72.90
Rolling Forcing†	10 s	98.64	98.89	93.67	94.73	98.83	62.91	72.93
Rolling Forcing†	20 s	98.71	98.86	93.67	90.41	96.90	59.90	**73.16**
Causal Forcing†	5 s	96.78	97.74	95.86	95.41	98.10	56.78	70.35
Mask Forcing†‡	5 s	98.83	99.09	95.37	95.57	98.86	63.30	73.11
StreamCeption	5 s	99.01	99.15	**97.91**	**97.40**	**99.36**	61.89	71.27
StreamCeption	10 s	98.77	98.94	96.43	96.40	**99.36**	61.04	71.17
StreamCeption	20 s	98.77	98.93	96.42	96.40	99.35	61.06	71.17

Evaluation metrics. We report seven dimensions from VBench (Huang et al., 2024) and its I2V evaluation suite, covering temporal consistency, frame quality, and fidelity to the conditioning image. Subject consistency (SC) and background consistency (BC) measure the stability of subject appearance and background content across generated frames, while motion smoothness (MS) evaluates temporal smoothness of motion. Aesthetic quality (AQ) and imaging quality (IQ) assess the aesthetic appeal and technical quality of individual frames. I2V subject consistency (I2V-S) and I2V background consistency (I2V-B) measure alignment with the input image's subject and background, respectively. They complement SC and BC by evaluating preservation of the visual condition rather than consistency only within the generated sequence. All scores are multiplied by 100 and reported to two decimal places; higher values indicate better performance. Bold denotes the highest displayed value in each metric column, including ties.

4.1 Experimental Results

Comparison at matched durations. At 5 seconds, StreamCeption achieves the highest SC, BC, and MS among the compared methods, exceeding the strongest forcing baseline in each dimension (Causal Forcing for SC, Mask Forcing for BC and MS) by 2.05, 1.83, and 0.50 points (Table 1). Its I2V-S and I2V-B also exceed all forcing baselines and remain close to Wan2.2.

At 10 and 20 seconds, StreamCeption surpasses Rolling Forcing across all five consistency and smoothness dimensions. The SC, BC, and MS margins are 2.76, 1.67, and 0.53 points at 10 seconds. At 20 seconds, the BC and MS margins widen to 5.99 and 2.45 points, while the SC margin remains stable at 2.75 points. AQ is also higher at 20 seconds (61.06 versus 59.90). Fig. 4 illustrates differences in color consistency, action complexity, and motion naturalness.

Across the displayed results, Wan2.2 achieves the highest AQ (64.01), while Rolling Forcing at 20 seconds has the highest IQ (73.16). IQ averages independent frame-quality predictions without explicitly measuring temporal consistency or penalizing longer duration Huang et al. (2024). This high IQ can therefore coexist with the observed declines in BC and MS as generation extends. StreamCeption's lower IQ (71.17 at 20 seconds) indicates room for improving individual-frame quality alongside its stronger temporal consistency.

Generalization to longer durations. We further examine how StreamCeption's performance changes as generation extends from 5 to 10 and 20 seconds. The 20-second setting is approximately six times the duration of the 3.4-second training clips. Some appearance drift remains between 5 and 10 seconds: SC decreases by 1.48 points and BC by 1.00 point. Further extension to 20 seconds produces little additional change, with every reported score differing from its 10-second counterpart by at most 0.02 points. At 20 seconds, StreamCeption retains SC and BC scores of 96.42 and 96.40, MS of 99.35, and I2V-S and I2V-B of 98.77 and 98.93. These results support generation beyond the training duration using a bounded denoising context, with limited additional degradation in the measured dimensions between 10 and 20 seconds.

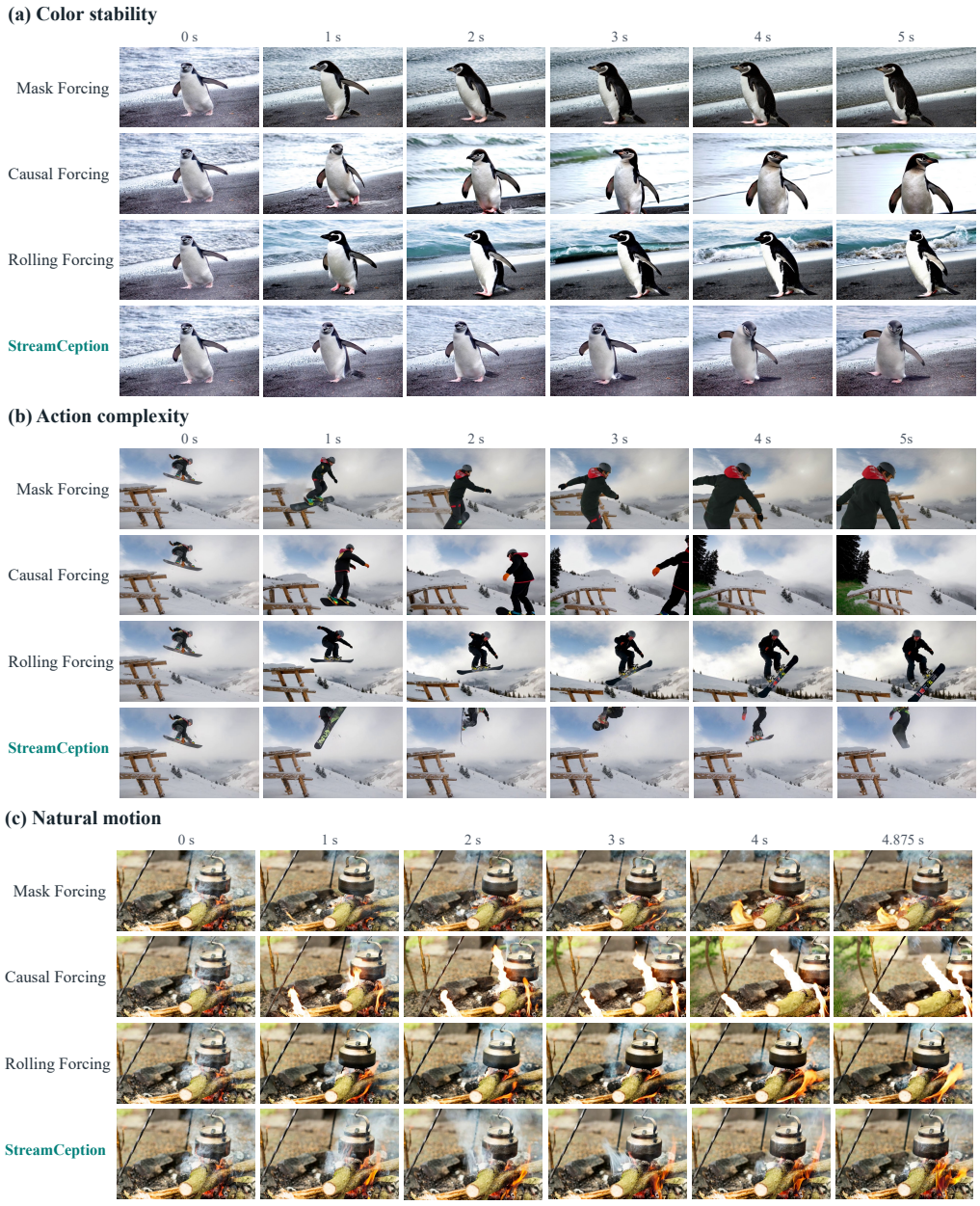


Figure 4: **Qualitative comparison of video generation methods.** (a) Color consistency, highlighting color shifts in the background sea. (b) Action complexity, illustrated by snowboard rotation during a jump. (c) Motion naturalness, reflected in drifting smoke and flickering flames. Frames are sampled at matched timestamps across methods.

5 Conclusion

We presented StreamCeption, which uses later context to selectively repair fully denoised internal latents within a bounded window while preserving the previously generated output video. A binary head selects fixed-strength re-noising, and bidirectional attention guides correction. Shared repair trials supervise selection through reconstruction improvement and train correction alongside flow matching. Experiments show competitive quality against the evaluated streaming baselines, with quantitative evaluation up to 20 seconds and qualitative examples extending to one minute. These results support internal context refinement for mitigating error propagation in streaming generation.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing of author-written text. We have not used generative AI tools for ideation, method design, implementation, data collection, or analysis, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for grammar and style suggestions on selected passages. We have reviewed all AI-assisted work. Specifically, all AI-suggested edits were reviewed by the authors for technical accuracy, terminological consistency, and preservation of meaning; no AI-generated text was incorporated without manual verification. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work studies streaming video generation from an initial image and a text prompt. The method operates on latent representations produced by a pretrained video VAE and does not introduce a new data collection procedure or a new dataset. All training and evaluation data are existing video datasets used by the base model and the VBench-I2V benchmark (Huang et al., 2024). Thus, we do not need to state ethics concerns.

REPRODUCIBILITY STATEMENT

We describe the model architecture, streaming schedule, repair controller, and training objectives in Sec. 3. Reference hyperparameters, including the window capacity W, retained-context length P, startup and look-ahead counts, inference grid size N, SNR shift, repair thresholds, context-development budget D, attempt budget $A_{\max}$, re-noising strength s_{rep}, and loss weights λ_{cls} and λ_{rep}, are stated in Secs. 3.1–3.3. The inference ordering, the bounded repair controller, and the state-restoration procedure are specified in Appendix A.2. The training construction, gradient paths, and loss normalization are given in Appendix A.3.

Evaluation uses the publicly available VBench-I2V benchmark (Huang et al., 2024) with the protocol described in Sec. 4. We report all metrics without cherry-picking and include an ablation of selective repair in Sec. A.5. The base model and the compared streaming methods are referenced through their public implementations; any compatibility modifications applied to a baseline are noted in the corresponding table caption.

We plan to release the code, configuration files, and the sampling scripts used to produce the reported results as supplementary material. No new dataset or pretrained checkpoint is introduced by this work.

REFERENCES

AlayaWorld Team, Kaipeng Zhang, Chuanhao Li, Yifan Zhan, Yongtao Ge, Yuanyang Yin, Jiaming Tan, Kang He, Liaoyuan Fan, Ruicong Liu, Xiaojie Xu, Xuangeng Chu, Zhen Li, Zhengyuan Lin, Zhixiang Wang, Zian Meng, and Zihui Gao. AlayaWorld: Long-Horizon and Playable Video World Generation, 2026. URL https://arxiv.org/abs/2607.06291.

Tariq Berrada Ifriqi, John Nguyen, Karteek Alahari, Jakob Verbeek, and Ricky T. Q. Chen. Flowception: Temporally expansive flow matching for video generation, 2025. URL https://arxiv.org/abs/2512.11438.

Tri Cao, Hung Nguyen, Phong Nguyen, and Khoi Nguyen. Recency forcing: Bridging the long-horizon gap in autoregressive video generation, 2026. URL https://arxiv.org/abs/2609.19729.

Haoxuan Che, Xuanhua He, Quande Liu, Cheng Jin, and Hao Chen. Gamegen-x: Interactive open-world game video generation. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8VG8tpPZhe.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion, 2024. URL https://arxiv.org/abs/2407.01392.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

Yuwei Guo, Ceyuan Yang, Hao He, Yang Zhao, Meng Wei, Zhenheng Yang, Weilin Huang, and Dahua Lin. End-to-End Training for Autoregressive Video Diffusion via Self-Resampling, 2025. URL https://arxiv.org/abs/2512.15702.

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, Eitan Richardson, Guy Shiran, Itay Chachy, Jonathan Chetboun, Michael Finkelson, Michael Kupchick, Nir Zabari, Nitzan Guetta, Noa Kotler, Ofir Bibi, Ori Gordon, Poriya Panet, Roi Benita, Shahar Armon, Victor Kulikov, Yaron Inger, Yonatan Shiftan, Zeev Melumian, and Zeev Farbman. LTX-2: Efficient Joint Audio-Visual Foundation Model, 2026. URL https://arxiv.org/abs/2601.03233.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020. URL https://arxiv.org/abs/2006.11239.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. URL https://arxiv.org/abs/2204.03458.

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. CogVideo: Large-scale Pretraining for Text-to-Video Generation via Transformers, 2022. URL https://arxiv.org/abs/2205.15868.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the train-test gap in autoregressive video diffusion, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Akio Kodaira, Tingbo Hou, Ji Hou, Markos Georgopoulos, Felix Juefei-Xu, Masayoshi Tomizuka, and Yue Zhao. StreamDiT: Real-time streaming text-to-video generation, 2025. URL https://arxiv.org/abs/2507.03745.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. HunyuanVideo: A Systematic Framework For Large Video Generative Models, 2024. URL https://arxiv.org/abs/2412.03603.

Wuyang Li, Wentao Pan, Po-Chien Luan, Yang Gao, and Alexandre Alahi. Stable Video Infinity: Infinite-Length Video Generation with Error Recycling, 2025. URL https://arxiv.org/abs/2510.09212.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling, 2022. 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, 2025. URL https://arxiv.org/abs/2509.25161.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations, 2021. URL https://arxiv.org/abs/2108.01073.

PAN Team, Zihan Liu, Yi Gu, Mingkai Deng, Guangyi Liu, Zeyu Feng, Qiyue Gao, Yiyan Hu, Benhao Huang, Yichi Yang, Kun Zhou, Jiannan Xiang, Zhiting Hu, Zhengzhong Liu, and Eric P. Xing. PAN: A World Model for General, Actionable, and Long-Horizon World Simulation, 2025. URL https://arxiv.org/abs/2511.09057.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, 2022. URL https://arxiv.org/abs/2212.09748.

Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, Yihang Chen, Jie Liu, Yansong Cheng, Yao Yao, Jiayi Zhu, Yihao Meng, Kecheng Zheng, Qingyan Bai, Jingye Chen, Zehong Shen, Yue Yu, Xing Zhu, Yujun Shen, and Hao Ouyang. Advancing Open-source World Models, 2026. URL https://arxiv.org/abs/2601.20540.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, 2021. URL https://arxiv.org/abs/2112.10752.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling Diffusion Models, 2024. URL https://arxiv.org/abs/2402.09470.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, 2020. URL https://arxiv.org/abs/2011.13456.

WAN, 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, 2025. URL https://arxiv.org/abs/2503.20314.

Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive Autoregressive Video Diffusion Models, 2024. URL https://arxiv.org/abs/2410.08151.

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, 2024. URL https://arxiv.org/abs/2412.07772.

Zhengyang Yu, Akio Hayakawa, Masato Ishii, Qingtao Yu, Takashi Shibuya, Jing Zhang, and Yuki Mitsufuji. AutoRefiner: Improving Autoregressive Video Diffusion Models via Reflective Refinement Over the Stochastic Sampling Path, 2025. URL https://arxiv.org/abs/2512.11203.

Zhuoran Zhao, Shengju Qian, Tongtong Liang, Xianghao Kong, Songchun Zhang, Junchao Huang, Guian Fang, Xin Wang, Pan Hui, and Anyi Rao. Mask forcing: Improving autoregressive video diffusion distillation via dual-noise masking rollout, 2026. URL https://arxiv.org/abs/2609.09123.

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, 2026a. URL https://arxiv.org/abs/2602.02214.

Yueting Zhu, Yuehao Song, Kaicheng Zhang, Bao Tang, Shaoyu Chen, Qian Zhang, Wenyu Liu, and Xinggang Wang. Stream forcing: Constructing unified training trajectory for robust streaming video generation, 2026b. URL https://arxiv.org/abs/2608.10439.

A Appendix

A.1 Related Work

Diffusion and flow-based generative models. Diffusion models learn to reverse a gradual corruption process, enabling sample generation through iterative denoising (Ho et al., 2020; Song et al., 2020). Latent diffusion reduces the computational cost of high-resolution generation by operating in an autoencoder's latent space (Rombach et al., 2021), while diffusion transformers provide a scalable architecture for modeling such representations (Peebles & Xie, 2022). Flow matching instead learns continuous generative trajectories by regressing velocity fields along conditional probability paths (Lipman et al., 2022), and provides the generative formulation used in StreamCeption. Re-noising an existing sample followed by further denoising is also a well-established operation. For example, SDEdit (Meng et al., 2021) injects noise into an input and denoises it under a generative prior to trade off input preservation and synthesis. In StreamCeption, re-noising serves a different role: it selectively reopens the denoising trajectory of an internal video latent so that subsequently developed temporal context can refine its value.

Video generation and world models. Conditional video generation synthesizes temporal sequences from inputs such as text descriptions or initial images. CogVideo (Hong et al., 2022) and HunyuanVideo (Kong et al., 2024) study large-scale text-to-video generation, while Wan (WAN et al., 2025) supports both text-to-video and image-to-video synthesis. LTX-2 (HaCohen et al., 2026) further develops a unified foundation model for synchronized audio-visual generation. Recent video-based world models extend these capabilities toward temporally evolving and interactive environments. PAN (PAN Team et al., 2025) combines latent world-state prediction with a video diffusion decoder for action-conditioned simulation. LingBot-World (Robbyant Team et al., 2026) develops an open video-based simulator for extended interactive generation, while Alaya-World (AlayaWorld Team et al., 2026) targets playable generative worlds. These settings place increasing emphasis on preserving subject appearance, scene structure, and temporal coherence as generation proceeds beyond short pretrained video horizons. StreamCeption studies this underlying continuation problem at the streaming generation level, focusing on how errors in generated latent history can be corrected before they propagate further.

Streaming video generation. A major line of work generates frames or video chunks chronologically while conditioning on previously generated content. Diffusion Forcing (Chen et al., 2024) introduces independent per-token noise levels for causal sequence generation. CausVid (Yin et al., 2024) distills a bidirectional video diffusion model into a few-step causal generator. Self Forcing (Huang et al., 2025) exposes the model during training to its own generated history to reduce the train–test mismatch, while Resampling Forcing (Guo et al., 2025) constructs imperfect history through self-resampling for end-to-end training. Stable Video Infinity (Li et al., 2025) similarly recycles model-generated errors during fine-tuning to improve recovery from imperfect conditioning. More recent forcing-based methods further improve robustness through specialized autoregressive training trajectories (Liu et al., 2025; Zhu et al., 2026a; Zhao et al., 2026). Collectively, these approaches primarily improve how a generator predicts new content when its conditioning history is imperfect.

A complementary direction overlaps denoising across neighboring temporal positions. Rolling Diffusion (Ruhe et al., 2024) and Progressive Autoregressive Video Diffusion (Xie et al., 2024) assign different denoising stages to consecutive frames and advance a sliding temporal window. Rolling Forcing (Liu et al., 2025) and StreamDiT (Kodaira et al., 2025) retain bidirectional communication among latents inside the active window while maintaining chronological video generation. Flowception (Berrada Ifriqi et al., 2025) further formulates video generation over variable-length sequences, coupling frame insertion with continuous denoising and individual per-frame times. StreamCeption inherits the asynchronous per-latent denoising representation and bidirectional context exchange, but replaces arbitrary temporal insertion with consecutive latent appending in a bounded streaming window.

Refinement of generated video states. Several methods revisit intermediate generated states rather than treating every prediction as final. AutoRefiner (Yu et al., 2025), for example, learns to refine noise along stochastic sampling trajectories using reflective history features. StreamCep-

tion addresses a distinct point in the generation process: residual errors that remain after a latent has already completed its initial denoising trajectory. A completed latent is temporarily retained inside the bounded working window while later latents continue to develop. The resulting temporal context is used by a learned binary head to decide whether the retained latent should remain unchanged or be revisited. When repair is selected, the latent is re-noised at a fixed strength and refined through bidirectional attention. The repaired working latent then serves as internal conditioning for subsequent generation, while previously output video remains immutable. Our central distinction is therefore selective post-denoising repair of internal conditioning history, together with explicit supervision of both the keep-or-repair decision and the resulting correction.

A.2 Inference

This section specifies the inference operations summarized in Fig. 3 and Secs. 3.1-3.2. We distinguish internal context refinement from chronological video output throughout. Algorithms 1 and 2 separate the streaming loop from the bounded repair controller. We use *Hold* for developing later context while keeping the completed candidate unchanged, and *Review* for reassessing it with a fresh decision forward.

State and output conventions. The internal window is $\mathcal{W} = \mathcal{H} \| \mathcal{B}$, where

$$\mathcal{H} = [(\mathbf{h}_i, 0)]_{i=\max(1,n-P)}^{n-1}, \qquad \mathcal{B} = [(\mathbf{x}_i, \sigma_i)]_{i=n}^{m}. \tag{13}$$

Here n is the oldest active position eligible for internal repair, m the latest initialized position, and $1 \le P < W$ the retained-context length, with $|\mathcal{H}| \le P$ and $|\mathcal{W}| \le W$. The retained values $\mathbf{h}_i$ are accepted internal latents that have left $\mathcal{B}$ and stay fixed while they reside in $\mathcal{H}$. They may differ from their immutable output snapshots $\widehat{\mathbf{x}}_i$ after internal repair. The separate output position r indexes the first latent not yet copied into $\mathcal{O} = [\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_{r-1}]$. The scalar r differs from the bold image-reference latent $\mathbf{r}$. The sampler maintains $n \le r$: position n is the internal repair candidate, while position r supplies the next new output. For $n \le i < r$, the active value $\mathbf{x}_i$ is an internal copy of a position already output. Its repair may change $\mathbf{x}_i$ and eventually $\mathbf{h}_i$, but never $\widehat{\mathbf{x}}_i$ or previously displayed RGB frames. Our streaming experiments include $n < r$; the offset $r = n + 3$ in Fig. 3(b) is an illustration, not a fixed relation imposed on every sampler state.

Consecutive appending and noise schedules. Initially, $\mathcal{O} = [\mathbf{r}]$, $\mathcal{H} = [(\mathbf{r}, 0)]$, $\mathcal{B} = \varnothing$, $n = r = 2$, and $m = 1$. An append operation adds exactly the next temporal position:

$$\boldsymbol{\epsilon}_{m+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathcal{B} \leftarrow \mathcal{B} \| [(\boldsymbol{\epsilon}_{m+1}, 1)], \quad m \leftarrow m + 1. \tag{14}$$

Appending requires $m < T$ and $|\mathcal{W}| < W$. The fixed startup and look-ahead settings control initialization and replenishment of the active window. The reference configuration uses startup and look-ahead counts of 16 each, window capacity $W = 21$, and retained-context length $P = 4$. For example, successive append and denoising operations can yield

$$\begin{aligned} &[(\boldsymbol{\epsilon}_2, 1)] \\ &\quad \longrightarrow [(\mathbf{x}_2, \sigma_2), (\boldsymbol{\epsilon}_3, 1)] \\ &\quad \longrightarrow [(\mathbf{x}_2, 0), (\mathbf{x}_3, \sigma_3), (\boldsymbol{\epsilon}_4, 1)]. \end{aligned} \tag{15}$$

Here the intermediate σ_2 and σ_3 lie in $(0, 1)$; each arrow can include multiple solver updates.

Each active latent follows its own position on the shared decreasing grid $\mathcal{G} = \{s_0, \ldots, s_N\}$, where $1 = s_0 > \cdots > s_N = 0$. The reference configuration uses the Wan noise scheduler with $N = 50$ and SNR shift 5.0. A forward evaluates the current context $\mathcal{C} = (\mathbf{r}, \mathcal{H}, \mathcal{B}, \mathbf{c})$, including temporal positions and individual noise levels. Let $\mathbf{v}_i$ denote the velocity supplied by the current forward. For each unfinished active latent, the next lower grid level σ_i^+ determines its Euler update:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + (\sigma_i^+ - \sigma_i)\mathbf{v}_i, \qquad \sigma_i \leftarrow \sigma_i^+. \tag{16}$$

All velocities in one update use the same input state. Ordinary updates leave completed latents, retained context, and the persistent reference unchanged. In ordinary generation, $\mathbf{v}_i$ uses the chosen classifier-free guidance. In the Review and repair forwards, $\mathbf{v}_i$ is the conditional model velocity $\mathbf{u}_{\theta,i}$ evaluated on the corresponding input context; the repair score and endpoint also use this conditional prediction.

Algorithm 1 Streaming Generation with Separate Internal Context and Output

Input: Model $\mathbf{u}_\theta$, repair head g_ψ, reference $\mathbf{r}$, text $\mathbf{c}$, target T; grid $\mathcal{G}$, append schedule, W, P, D, and repair settings $(e, \tau, \tau_{\text{acc}}, s_{\text{rep}}, A_{\max})$.

$\mathcal{O} \leftarrow [\mathbf{r}]$, $\mathcal{H} \leftarrow [(\mathbf{r}, 0)]$, $\mathcal{B} \leftarrow \varnothing$; $n \leftarrow 2$, $r \leftarrow 2$, $m \leftarrow 1$.
while $r \leq T$ **do**
  Append scheduled consecutive noise latents subject to capacity and $m < T$; initialize n if $\mathcal{B}$ is empty.
  if $\sigma_n > 0$ **then**
    Advance all unfinished active latents using Eq. 16.
    OUTPUTREADY; **continue**.
  end if
  for up to D updates while unfinished later latents remain **do**
    Hold: keep $\mathbf{x}_n$ fixed; advance unfinished later latents using Eq. 16.
    OUTPUTREADY.
  end for
  if $r > T$ **then**
    break.
  end if
  $(\mathcal{B}, m) \leftarrow$ REPAIRCONTEXT$(\mathcal{B}, m, n)$. ▷ Algorithm 2
  OUTPUTREADY.
  $\mathbf{h}_n \leftarrow \text{copy}(\mathbf{x}_n)$; move $(\mathbf{h}_n, 0)$ into $\mathcal{H}$ and remove n from $\mathcal{B}$; retain up to P recent context values.
  $n \leftarrow n + 1$.
end while
return $\mathcal{O}$.

procedure OUTPUTREADY
  while $r \leq m$ **and** $\sigma_r = 0$ **do**
    $\widehat{\mathbf{x}}_r \leftarrow \text{copy}(\mathbf{x}_r)$; append $\widehat{\mathbf{x}}_r$ to $\mathcal{O}$.
    Supply only this new snapshot to the chronological VAE decoding; $r \leftarrow r + 1$.
  end while
end procedure

Chronological output and internal window advancement. OUTPUTREADY copies consecutive completed latents starting at the next output position r into $\mathcal{O}$, advancing r after each copy. It never reads an earlier output position again. The VAE consumes these immutable snapshots in temporal order and appends new RGB frames, with decoder chunking handled separately. The output decoder uses its existing output history or cache; internal repair does not replace that history or rewind the cache.

New output is produced at completed sampler states, after any ongoing repair controller has returned its retained internal state. Algorithm 1 makes these boundaries explicit: ordinary denoising and context development can supply new output; the decision, repair retries, and any fallback finish before the next output step. The output prefix and position r are never part of a repair snapshot. After position n has both an output snapshot and an accepted internal value, the internal value moves into $\mathcal{H}$. This advances n and frees capacity without changing earlier output.

Hold: developing later context. Once the oldest internal candidate reaches $\sigma_n = 0$, its value $\mathbf{y}_n = \mathbf{x}_n$ is held unchanged while unfinished later latents undergo up to D additional updates. This wait occurs once per internal candidate and is not repeated on retries. The reference configuration uses $D = 1$. The Hold/Review row pairs in Fig. 3(b) summarize these stages together with the intervening updates, rather than individual transformer evaluations.

Review: the decision forward. A fresh conditional forward evaluates the held candidate under the developed context at $\sigma_n = 0$:

$$\mathbf{p}_n = \text{softmax}\left[g_\psi \left(\frac{1}{S} \sum_{q=1}^{S} \mathbf{h}_{\theta,n,q}(\mathcal{C}) \right) \right], \qquad \rho_n = p_{n,1}. \tag{17}$$

Here S is the number of spatial tokens per latent, and $\mathbf{h}_{\theta,n,q}$ is a hidden feature of the decision forward, distinct from the retained VAE latent $\mathbf{h}_i$. Class 0 denotes *keep* and class 1 denotes *repair*. The same forward advances unfinished later latents once, regardless of its decision. Repair uses the context after this update.

Algorithm 2 Bounded Internal Repair: REPAIRCONTEXT

Input: Active suffix $\mathcal{B}$, latest position m, completed candidate n; context and repair settings from Algorithm 1.

$a_n \leftarrow 0$, snapshot bank $\mathcal{S} \leftarrow \varnothing$.
loop
  Review: run a fresh conditional decision forward at $\sigma_n = 0$ to obtain velocities and ρ_n.
  Save $(a_n, \rho_n, \mathrm{copy}(\mathcal{B}), m)$ in $\mathcal{S}$, including all active noise levels.
  Advance unfinished later latents once with this forward.
  $\tau_n \leftarrow \tau$ if $a_n = 0$; otherwise $\tau_n \leftarrow \tau_{\mathrm{acc}}$.
  if $e = 0$ **or** $\rho_n \le \tau_n$ **then**
    return $(\mathcal{B}, m)$. ▷ Retain current internal value
  else if $a_n = A_{\max}$ **then**
    Select a snapshot in $\mathcal{S}$ with minimum saved repair score and restore $(\mathcal{B}, m)$.
    return $(\mathcal{B}, m)$. ▷ Restore working state only
  end if
  Set $\mathbf{y}_n \leftarrow \mathbf{x}_n$; re-noise candidate n using Eq. 7; set $\sigma_n \leftarrow s_{\mathrm{rep}}$.
  Run a conditional repair forward on the updated window.
  Set $\mathbf{x}_n \leftarrow \mathbf{x}_n^{\mathrm{rep}}$ using Eq. 8, and $\sigma_n \leftarrow 0$.
  Advance unfinished neighbors using the same forward and their ordinary Euler updates.
  $a_n \leftarrow a_n + 1$.
end loop

Repair execution and acceptance. The strength s_{rep} is the nearest positive inference-grid level to the configured strength (nominally 0.10). The head makes a binary decision and does not predict this strength. Before repair, the trigger threshold is τ; after at least one completed repair, the acceptance threshold is $\tau_{\mathrm{acc}} \in [\tau, 1]$. With completed-attempt count a_n, set $\tau_n = \tau$ if $a_n = 0$ and $\tau_n = \tau_{\mathrm{acc}}$ otherwise. A repair event satisfies

$$E_n = e\,\mathbf{1}[\sigma_n = 0]\,\mathbf{1}[\rho_n > \tau_n]\,\mathbf{1}[a_n < A_{\max}]. \tag{18}$$

Here $e \in \{0, 1\}$ enables repair, and $\mathbf{1}[A]$ equals 1 when condition A holds and 0 otherwise. Only the current internal candidate n is eligible. Equations 7 and 8 implement one conditional endpoint prediction per repair. That forward also advances unfinished neighbors on their ordinary clocks; completed neighbors remain fixed. A fresh Review forward reassesses the repaired candidate at $\sigma_n = 0$, without repeating Hold.

The reference configuration is $\tau = 0.5$, $\tau_{\mathrm{acc}} = 1$, and $A_{\max} = 1$. Thus, an initial score above 0.5 triggers one repair, and its completed correction is retained at the subsequent Review for any finite probability score, including $\rho_n = 1$. This accepts a completed correction by design; it does not certify that the repaired latent has attained a particular visual quality. The algorithm below also covers other thresholds and retry budgets.

An accepted candidate is retained without competing against earlier snapshots. Setting $\tau_{\mathrm{acc}} = \tau$ uses the same criterion before and after repair, while $\tau_{\mathrm{acc}} = 1$ retains the first completed repair with a finite score, provided it was triggered and the budget permitted it. The acceptance threshold trades further refinement against retaining the current correction; $A_{\max}$ caps the additional repair work. Neither setting changes the initial trigger.

Fallback and state restoration. In the general controller, fallback occurs only when another repair is requested after budget exhaustion. It is not reached for finite probability scores under the reference settings above. The bank includes the original completed candidate ($k = 0$) and every evaluated repair ($k = 1, \ldots, a_n$). We select

$$k^\star \in \arg\min_{0 \le k \le a_n} \rho_n^{(k)}. \tag{19}$$

Thus, the original, an intermediate, or the final candidate can be retained. The rule neither always restores the original nor always retains the last repair. It reuses existing evaluations as a stopping heuristic, avoiding additional repair or a separate quality evaluator. The score measures remaining repair preference, not reconstruction accuracy, and each score belongs to its own temporal context.

Snapshots are taken before the decision forward advances neighbors. Restoring the whole active suffix and its clocks keeps the selected candidate paired with the context under which it was evaluated. Retained context $\mathcal{H}$, the output prefix $\mathcal{O}$, output index r, and decoder state are excluded

from restoration. No new output snapshot is created while the repair controller is still choosing its retained state. Once it returns, OUTPUTREADY resumes at the unchanged position r.

Boundary cases and cost. No position beyond T is initialized. With $T = 1$, the initial-image latent is the complete output. Context development is skipped if $D = 0$ or no unfinished later latent remains. Repair-on and repair-off use the same context-development budget; disabling repair changes only the repair branch. With $A_{\max} = 0$, a positive repair request reaches fallback immediately, whose sole candidate is $k = 0$. Sampling terminates after all T output snapshots have been stored; further internal repair then cannot affect new output. Inference uses no clean targets.

Each transformer forward processes at most W window latents and one persistent reference. With S spatial tokens per latent, dense attention costs $\mathcal{O}(((W+1)S)^2)$ per forward. Per internal candidate, the controller uses at most D Hold updates, $A_{\max}+1$ Review forwards, and $A_{\max}$ repair forwards, in addition to ordinary denoising. At most $A_{\max}+1$ evaluated active states are retained in the snapshot bank. Working context and snapshot storage are bounded independently of T, while total sampling work and storage of the complete output grow with video duration.

A.3 REPAIR TRAINING DETAILS

This section expands the training construction in Sec. 3.3. MSE averages squared error over channels and spatial positions of a normalized VAE latent; CE is two-class cross-entropy, and we compute repair outcomes in FP32.

Reconstruction proposals. A training clip provides clean latents $\mathbf{x}^\star_{1:L}$ and the first-image reference. We sample a temporal window and per-latent noise levels following the streaming schedule, including context-development delays at preceding repair positions. The selected probe j is constructed before its own context wait, so that this delay is applied only once. We partially noise its clean latent:

$$\mathbf{z}_j = (1-s)\mathbf{x}^\star_j + s\boldsymbol{\epsilon}_j, \qquad \boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{20}$$

The reference configuration samples s from inference-grid levels in $[0.10, 0.30]$. With surrounding latents and conditioning fixed, the current model follows all remaining solver steps to reconstruct $\mathbf{y}_j$, without gradients. We place it in the active window at $\sigma_j = 0$ and retain $\mathbf{x}^\star_j$ as its target. The proposal level s and fixed repair level s_{rep} serve different purposes. These local reconstructions approximate free-running errors; they do not require clean targets at inference.

Native flow matching and Hold. Before developing later context, a native forward evaluates the window containing $\mathbf{y}_j$. Native flow matching applies only to valid noisy states, retaining the original ground-truth noising construction from Eq. 1:

$$\mathcal{L}_{\text{FM}} = \frac{1}{|\mathcal{F}|}\sum_{i\in\mathcal{F}} \text{MSE}(\mathbf{u}_{\theta,i}(\mathcal{C}), \boldsymbol{\epsilon}_i - \mathbf{x}^\star_i)\,. \tag{21}$$

Retained context, the persistent reference, padding, and the completed probe are excluded from $\mathcal{F}$. We then enter Hold: $\mathbf{y}_j$ remains unchanged while unfinished later latents undergo up to D detached updates, matching the inference context-development procedure.

Shared repair trials and gradient paths. In Review, a fresh decision forward evaluates $\mathbf{y}_j$ under the developed context and predicts $\mathbf{p}_j$. Its detached velocities first advance unfinished later latents once. A repair trial then uses these updated neighbors and fresh Gaussian noise to compute $\mathbf{x}^{\text{rep}}_j$, matching the decision–repair ordering at inference. Every selected probe receives a trial regardless of the head's prediction.

The reward R_j, binary label b_j, and magnitude $|R_j|$ in Eqs. 9-10 are detached. The reward uses raw endpoint MSE; the factor s_{rep}^{-2} applies only to the reconstruction objective. Raw, unclipped magnitude weights are averaged over the number of trials, preserving reward scale even with a single probe. Exact ties have zero classification weight. Classification gradients flow through $\mathbf{p}_j$ into the head and trainable shared features. The score therefore reflects an improvement-weighted repair preference, not a calibrated estimate of current visual quality. Section A.6 analyzes this interpretation and its transfer to free-running generation and, for configurations with multiple attempts, repeated repair.

Every trial contributes to the endpoint reconstruction loss in Eq. 11, including harmful trials. These receive a keep label while still teaching recovery toward the clean target. The reconstruction loss differentiates through the repair forward. Proposal collection, context development, and the decision-step neighbor update remain detached. The same repaired endpoint supplies the reward and reconstruction supervision, without an extra forward for reward computation.

Reconstruction normalization. Define the correction velocity that recovers the clean target as $\mathbf{u}_j^{\star,\mathrm{rep}} = (\widetilde{\mathbf{y}}_j - \mathbf{x}_j^{\star})/s_{\mathrm{rep}}$. The endpoint update yields

$$\frac{\mathrm{MSE}(\mathbf{x}_j^{\mathrm{rep}}, \mathbf{x}_j^{\star})}{s_{\mathrm{rep}}^2} = \mathrm{MSE}\Big(\mathbf{u}_{\theta,j}(\widetilde{\mathcal{C}}), \mathbf{u}_j^{\star,\mathrm{rep}}\Big)\,. \tag{22}$$

This target includes the proposal's existing reconstruction error, unlike the native Gaussian-path target $\boldsymbol{\epsilon}_j - \mathbf{x}_j^{\star}$. Using unnormalized endpoint MSE would require the coefficient $\lambda_{\mathrm{rep}}/s_{\mathrm{rep}}^2$ to retain the same objective. Each loss is zero when its valid set is empty. The reference weights are $\lambda_{\mathrm{cls}} = 0.1$ and $\lambda_{\mathrm{rep}} = 1.0$.

A.4 Hyperparameter Settings

Table 2 summarizes the key sampling and training hyperparameters of StreamCeption. We distinguish the guidance used for ordinary generation from that used for repair, and specify the loss normalization associated with the reported training weights.

Table 2: Reference sampling and repair-training hyperparameters of StreamCeption. Repair uses conditional prediction; the reconstruction weight multiplies endpoint MSE normalized by s_{rep}^2.

Hyperparameter	Value
Sampling	
Window capacity W	21
Retained-context length P	4
Persistent image reference	Enabled
Startup / look-ahead counts	16 / 16
Ordinary denoising steps N	50
Noise-schedule shift	5.0
Ordinary generation CFG scale	2.0
Hold update budget D	1
Selective repair	
Repair target	Current internal candidate n
Repair head	Binary keep / repair
Trigger threshold τ	0.5
Acceptance threshold τ_{acc}	1.0
Maximum attempts $A_{\max}$	1
Nominal re-noising strength	0.10
Effective strength s_{rep}	Nearest positive grid level
Repair endpoint	One conditional prediction
Repair CFG scale	1.0 (conditional only)
Repair training	
Streaming-state sampling probability	0.8
Repair-probe sampling probability	0.5
Repair-trial probability per selected probe	1.0
Proposal noise-level range	$[0.10, 0.30]$
Classification loss weight λ_{cls}	0.1
Reconstruction loss weight λ_{rep}	1.0

Sampling and guidance. Ordinary video generation uses a CFG scale of 2.0 and a noise-schedule shift of 5.0. Repair uses the conditional model prediction without additional classifier-free guidance, corresponding to a CFG scale of 1.0. The repair score is also obtained from the conditional forward.

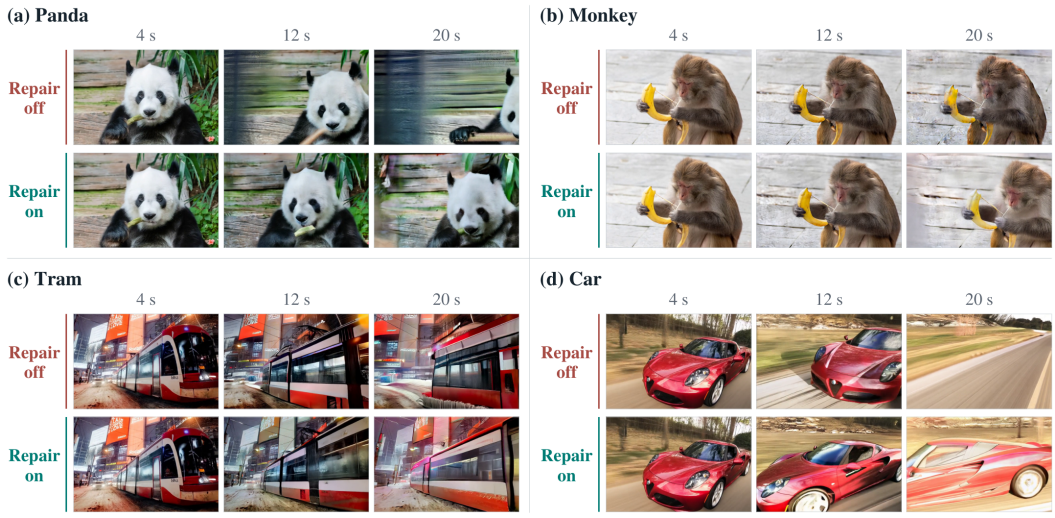


Figure 5: Ablation study of the repair process. Frames are sampled at 4, 12, and 20 seconds from four generated sequences. The two variants produce similar content at 4 seconds, but their trajectories diverge as generation continues. Repair is enabled in the bottom row of each pair.

Table 3: Ablation of selective repair at 10 seconds on I2V VBench. All other settings, including the context-development budget D, are identical between the two rows. Δ denotes repair-on minus repair-off.

Setting	I2V-S	I2V-B	SC	BC	MS	AQ	IQ
Repair off	97.79	97.95	95.68	95.32	99.35	60.26	71.12
Repair on	98.77	98.94	96.43	96.40	99.36	61.04	71.17
Δ	$+0.98$	$+0.99$	$+0.75$	$+1.08$	$+0.01$	$+0.78$	$+0.05$

Loss weighting. The native flow-matching objective has unit weight. λ_{cls} weights the repair classification objective, and λ_{rep} weights the repair reconstruction objective. The latter includes the normalization s_{rep}^{-2}, where s_{rep} is the effective repair noise level on the inference grid. Therefore, the reported reconstruction weight 1.0 applies to the normalized endpoint MSE. If raw endpoint MSE were used instead, the equivalent coefficient would be $\lambda_{\text{rep}}/s_{\text{rep}}^2$.

A.5 Ablation Study

Repair vs. No Repair. Table 3 isolates the effect of selective repair at 10 seconds. The two settings share the same context-development budget D and differ only in whether the repair branch is enabled. This compares the repair mechanism under the same prespecified context-development budget before the decision. The enabled branch includes the associated repair and Review forwards and their neighbor updates; the two settings do not enforce equal numbers of model evaluations. Enabling repair improves I2V-S, I2V-B, SC, and BC by 0.75–1.08 points. The largest gain appears in background consistency ($+1.08$) and image-conditioning fidelity ($+0.98$ and $+0.99$). Motion smoothness changes by $+0.01$, imaging quality increases by 0.05, and aesthetic quality increases by 0.78. The measured benefits therefore include stronger temporal consistency and input-image alignment, together with higher AQ, while MS and IQ change only slightly.

Figure 5 illustrates the same effect qualitatively. Frames are sampled at 4, 12, and 20 seconds from four generated sequences. The two variants produce visually similar content at 4 seconds, but their trajectories diverge as generation continues. The clearest differences appear in the panda and car examples. Without repair, the panda becomes partially displaced outside the frame, and the car is no longer visible at 20 seconds. With repair enabled, both subjects remain clearly visible, with the

panda's facial features and the car's overall appearance remaining recognizable. The monkey and tram examples also retain recognizable subject structure as pose and viewpoint change, although the differences between variants are less pronounced. These examples suggest that repairing active latents can help preserve subject continuity during extended generation, consistent with the motivation that refining the context used for subsequent generation can limit the propagation of earlier errors.

Repair operates on internal working latents within the active window, including copies of already output positions when $n < r$. The VAE receives only new output snapshots at position r; previously output video remains unchanged. The quantitative and qualitative results therefore reflect the same mechanism: a correction applied to internal history that affects subsequent generation without altering what the user has already seen.

A.6 Interpretation and Scope of Repair Supervision

This section distinguishes the learned repair preference from current latent quality and clarifies the scope of the reconstruction objective. The analysis describes the stated losses; it does not establish their calibration on free-running videos.

Meaning of the weighted classification score. For a fixed denoiser and training distribution, consider an ideal classifier conditioned on the decision input $\mathcal{C}$. The reward R in Eq. 9 varies with the original clean target associated with each training reconstruction and the sampled repair noise. Define $R_+ = \max(R, 0)$ and $(-R)_+ = \max(-R, 0)$. Let

$$A(\mathcal{C}) = \mathbb{E}[R_+ \mid \mathcal{C}], \qquad B(\mathcal{C}) = \mathbb{E}[(-R)_+ \mid \mathcal{C}]. \tag{23}$$

The conditional risk of the weighted cross-entropy in Eq. 10 is

$$\ell(\rho \mid \mathcal{C}) = -A(\mathcal{C}) \log \rho - B(\mathcal{C}) \log(1 - \rho). \tag{24}$$

When $0 < A(\mathcal{C}) + B(\mathcal{C}) < \infty$, its minimizer is

$$\rho^*(\mathcal{C}) = \frac{A(\mathcal{C})}{A(\mathcal{C}) + B(\mathcal{C})}. \tag{25}$$

If both expectations are zero, the classification loss supplies no preference. In general, ρ^* is an improvement-weighted action preference, not $\Pr(R > 0 \mid \mathcal{C})$ or an estimate of the candidate's reconstruction error.

Since $\mathbb{E}[R \mid \mathcal{C}] = A(\mathcal{C}) - B(\mathcal{C})$, $\rho^* > 1/2$ is equivalent to positive expected latent-MSE improvement. More generally,

$$\rho^*(\mathcal{C}) > \tau \quad \Longleftrightarrow \quad (1 - \tau)A(\mathcal{C}) > \tau B(\mathcal{C}). \tag{26}$$

For the reference trigger $\tau = 0.5$, this reduces to $A > B$. Larger trigger thresholds require a larger ratio of expected gain to expected degradation. This interpretation assumes the ideal classifier and the same conditional reward distribution at training and inference. It does not establish calibration of the learned head, and a fresh noise draw can still produce a harmful repair.

Limits of the fallback score. The minimum-score fallback in Algorithm 2 uses preference for another repair to choose an evaluated snapshot. That preference does not rank current quality. For example, consider an exhausted controller with $\tau = \tau_{\mathrm{acc}} = 0.5$ and two hypothetical candidates with MSE 0.30 and 0.03. Suppose each repair changes MSE by ± 0.01, with positive-reward probabilities 0.75 and 0.90, respectively. Their ideal scores are 0.75 and 0.90: both request another repair under this threshold, yet minimizing the score selects the higher-error candidate. Thus, the fallback provides bounded progress without ground truth, but its quality must be evaluated empirically; it does not select a provably best snapshot. This example concerns the general fallback rule. In the reference configuration, $\tau_{\mathrm{acc}} = 1$ accepts a completed repair with a finite score before it reaches fallback.

The correction target and its scale. Substituting the re-noised reconstruction into the target in Eq. 22 gives

$$\mathbf{u}_j^{\star,\mathrm{rep}} = \boldsymbol{\xi}_j - \mathbf{x}_j^\star + \frac{1 - s_{\mathrm{rep}}}{s_{\mathrm{rep}}}(\mathbf{y}_j - \mathbf{x}_j^\star). \tag{27}$$

The final term corrects the existing reconstruction residual. It is absent from the native Gaussian-path target and has coefficient 9 when the effective level is $s_{\mathrm{rep}} = 0.10$. For one trial with latent dimensionality d, the gradient of the normalized endpoint loss with respect to the predicted velocity is

$$\frac{\partial \ell_{\mathrm{rep}}}{\partial \mathbf{u}_{\theta,j}} = -\frac{2}{d\, s_{\mathrm{rep}}}(\mathbf{x}_j^{\mathrm{rep}} - \mathbf{x}_j^{\star}). \tag{28}$$

Large candidate errors or small repair levels can therefore increase the correction scale. The factor alone does not demonstrate unstable optimization. At fixed s_{rep}, the normalization is equivalent to weighting raw endpoint MSE by $\lambda_{\mathrm{rep}}/s_{\mathrm{rep}}^2$. Changing the repair level also changes the input distribution, so sensitivity to strength and loss weight requires empirical assessment rather than an algebraic stability claim.

Training scope and long-horizon claims. Training candidates are local reconstructions initialized from partially noised data, with neighbors held fixed during collection. Free-running candidates start from pure noise while neighbors evolve. The shared decision-step ordering aligns the repair procedure, but does not make these candidate distributions identical. Training uses one trial per selected candidate, and the reference inference configuration performs at most one repair per internal candidate. It reassesses the repaired endpoint in Review but retains it with $\tau_{\mathrm{acc}} = 1$. Configurations permitting further attempts use these post-repair scores to decide whether to repair again. Such successive-repair states are not directly sampled by the stated training construction. Transfer under these distribution shifts requires empirical evaluation. Finally, the reward measures the error of one latent against its original clean target. It is a training proxy, not a guarantee of perceptual quality, temporal consistency, or reduced long-horizon error accumulation. These effects require video-level evaluation.

A.7 Future Work

Future work will explore perceptual and temporal feedback to complement the current reconstruction-based repair supervision, including assessing how internal corrections influence subsequent generation. We will also investigate more efficient repair scheduling and opportunities to reuse intermediate computations, aiming to reduce inference overhead while preserving the consistency benefits of internal context refinement over longer streaming sequences.