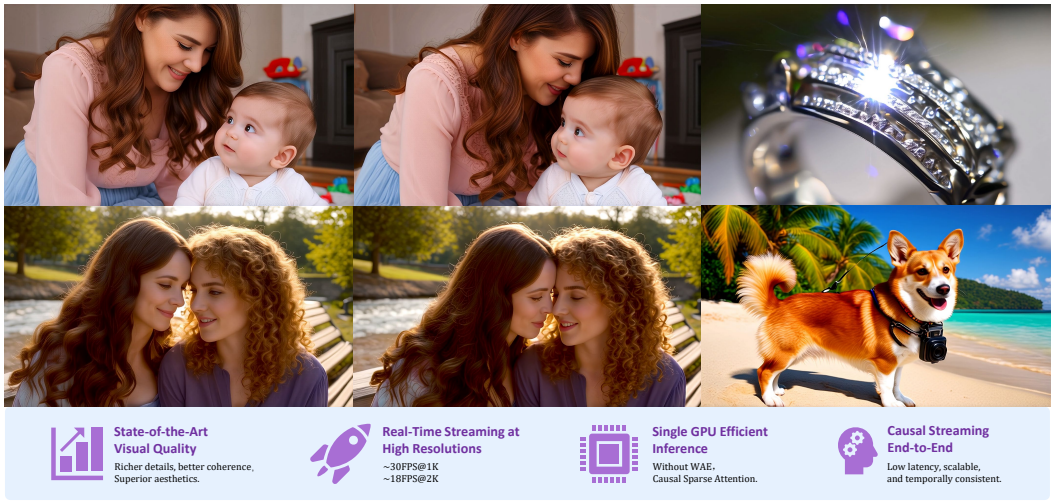
Ultra Flash: Scaling Real-Time Streaming Video Generation to High Resolutions

Luxury[1,2] **Jie Huang**[1,2] **Zihao Fan**[2] **Xiaoxiao Ma**[2]
Yuming Li[1] **Siming Fu**[1] **Jun-hao Zhuang**[1] **Zeyue Xue**[1]
Haoran Li[1] **Haoyang Huang**[1] **Nan Duan**[1]
[1]JD Explore Academy [2]University of Science and Technology of China



Abstract

While recent autoregressive video diffusion models achieve remarkable streaming quality, they remain confined to low resolutions (*e.g.*, 480P), leaving efficient, scalable, real-time high-resolution video generation a fundamental open challenge. To bridge this gap, we present Ultra Flash, a cascaded streaming framework capable of real-time high-resolution video generation. Ultra Flash achieves ∼30 FPS at 1K resolution and ∼18 FPS at 2K resolution on a single GPU through three key contributions: (1) an architecture-preserving T2V-to-TV2V super-resolution training paradigm coupled with an AIGC-oriented data degradation pipeline that effectively preserves the generative capability of the base model, enabling enhanced high-resolution detail when cascaded after mainstream low-resolution generative models; (2) a causal streaming latent upsampler paired with a high-resolution decoder, which enhances spatiotemporal coherence while enabling efficient latent spatial scaling and precise high-resolution decoding with negligible computational overhead; and (3) a cascade high-resolution streaming video generation optimization scheme that first performs hybrid-reward-enhanced sparse causalization and single-step distillation of the super-resolution model, then introduces cascaded streaming self-forcing preference optimization with dynamic cache management, jointly enhancing overall coherence, improving quality, and enabling real-time high-resolution streaming video generation. Extensive experiments demonstrate that Ultra Flash reliably produces ultra-high-resolution streaming video while maintaining state-of-the-art visual quality and superior efficiency.

Project Page: `https://xin1u.github.io/UltraFlash/`

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

1 Introduction

Video diffusion models have made extraordinary progress in generating photorealistic video from text prompts Team Wan et al. (2025); Polyak et al. (2024); Yang et al. (2024); Kong et al. (2024). Meanwhile, interactive applications—such as real-time previewing, game asset generation, and live content creation—demand *streaming* output at high resolution and low latency. Recent autoregressive adaptations Yin et al. (2025); Huang et al. (2025); Zhu et al. (2026) have taken a promising step by converting bidirectional DiTs into causal, chunk-wise generators via asymmetric distillation Yin et al. (2025) and self-forcing rollouts Huang et al. (2025), enabling real-time streaming at 480P on a single GPU. However, scaling these methods to high resolutions (*e.g.*, 1K or 2K) for practical deployment remains an open problem.

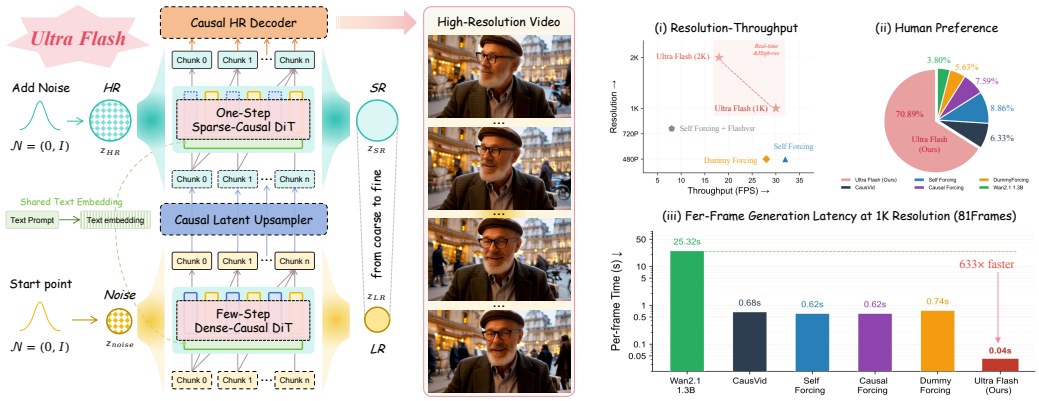
Since directly generating high-resolution video is prohibitively expensive, the cascaded paradigm Gao et al. (2025); Zhang et al. (2025b); Meituan LongCat Team et al. (2025)—first generating low-resolution video to capture semantics and motion, then upscaling via super-resolution (SR) to supplement high-frequency detail—has emerged as a practical solution. However, existing cascaded approaches suffer from several fundamental limitations. Some streaming diffusion-based SR methods Zhuang et al. (2025); Shiu et al. (2025) achieve high efficiency but operate in pixel space, introducing additional encode–decode overhead Zhang et al. (2025c) in the cascaded pipeline and requiring fundamental architectural modifications that forfeit the generative capability of the pre-trained T2V model, making training difficult. Other works attempt to address this through latent-space upsampling followed by a cascaded SR model FSVideo Team et al. (2026); Zhang et al. (2025b). However, their upsampling strategies either adopt naive interpolation Zhang et al. (2025b); SII-GAIR et al. (2026) or rely on large-scale upsampler models Wu et al. (2025); HaCohen et al. (2026). Both neglect the sensitivity of latent video representations to spatiotemporal consistency and cannot perform causal streaming extrapolation. Consequently, the subsequent SR stage must inject heavy noise to mitigate the frequency aliasing and spatiotemporal incoherence introduced by upsampling FSVideo Team et al. (2026). This heavy noise coverage over low-resolution information makes subsequent SR training—and its acceleration via distillation—considerably more difficult. Furthermore, existing high-resolution methods suffer from quadratic attention complexity and their SR components are not designed for one-step inference, making cascaded end-to-end optimization infeasible and leading to train-test inconsistency that compounds quality degradation.

To bridge this gap, we present Ultra Flash (Fig. 1), a cascaded streaming framework that scales real-time autoregressive video generation to high resolutions. Ultra Flash achieves $\sim$30 FPS at 1K resolution and $\sim$18 FPS at 2K resolution on a single GPU through three key contributions:

Efficient architecture-preserving T2V-to-TV2V SR training paradigm. We propose an efficient training paradigm that converts any pre-trained T2V model into a TV2V multimodal generative SR model without architectural modification Zhuang et al. (2025); Shiu et al. (2025), preserving the original generative capability. We further design an AIGC-oriented data degradation pipeline tailored to the characteristics of AI-generated video, effectively retaining model priors and enabling enhanced high-resolution detail when cascaded after mainstream low-resolution generative models.

Ultralight streaming latent upsampler with high-resolution decoder. We design a causal memory network that upsamples low-resolution latents to high resolution directly in latent space with temporal coherence. Unlike pixel-space VSR methods He et al. (2024); Zhuang et al. (2025) that introduce substantial overhead or latent cascaded approaches Zhang et al. (2025b); Wu et al. (2025) that rely on naive interpolation and restoration requiring heavy noise to mitigate aliasing, our spatiotemporally coherent upsampler adds $<5\%$ pipeline cost while eliminating the need for heavy noise injection, substantially reducing SR training and distillation difficulty. Paired with a ultralight high-resolution decoder, Ultra Flash enables efficient latent spatial scaling and precise high-resolution decoding, laying the foundation for high-resolution streaming generation.

Cascaded high-resolution streaming generation optimization. Building on the above models, we devise a comprehensive optimization scheme to enable real-time high-resolution streaming. First, we perform *hybrid-reward-enhanced sparse causalization and single-step distillation*: dynamic block-sparse causal attention Guo et al. (2024) replaces dense attention for streaming-compatible inference, while distribution matching distillation Liu et al. (2025) compresses multi-step denoising to a single step, with perceptual and aesthetic reward signals Xu et al. (2023); Ke et al. (2021); Zhang et al. (2026) to directly optimize for visual quality. Then, we introduce *cascaded streaming*



(a) Ultra Flash: Real-Time & High-Resolution streaming framework.

(b) Quality–speed comparison with prior methods.

Figure 1: (a) **Ultra Flash** framework. (b) Quality–speed comparison with prior methods. Ultra Flash scales to 1K and 2K resolution while achieving better quality and real-time throughput

self-forcing preference optimization with dynamic cache management: the low-resolution generator and the high-resolution SR model are jointly rolled out in a cascaded streaming fashion, where a preference optimization objective explicitly trains on self-generated context to close the train-test gap, while a dynamic cache management mechanism can further enhance the generation efficiency. Together, these designs jointly enhance overall temporal coherence, improve visual quality, and realize real-time high-resolution streaming video generation.

2 Method

Given any low-resolution autoregressive streaming generator (*e.g.*, Self Forcing Huang et al. (2025)), Ultra Flash cascades three components to scale real-time video generation to high resolutions (Fig. 2): (1) an *architecture-preserving T2V-to-TV2V SR training paradigm* with an AIGC-oriented degradation pipeline that converts the pre-trained T2V model into a generative super-resolution model without architectural modification (§2.1); (2) a *causal streaming latent upsampler* paired with a high-resolution decoder that lifts low-resolution latents to high resolution with spatiotemporal coherence (§2.2); and (3) a *cascaded high-resolution streaming optimization* scheme comprising sparse causalization, single-step distillation, self-forcing preference optimization, and dynamic cache management (§2.3).

2.1 Architecture-Preserving T2V-to-TV2V SR Training Paradigm

Existing pixel-space SR methods Zhuang et al. (2025); Shiu et al. (2025) require fundamental architectural modifications (*e.g.*, LQ projection layers, modified attention patterns) that forfeit the generative capability of the pre-trained T2V model. We instead propose a paradigm that repurposes any T2V model as a TV2V generative SR model *without architectural change*, preserving the full generative prior.

Conditioning Mechanism. The upsampled low-resolution latent $\mathbf{z}^{\text{HR}}$ from the streaming latent upsampler is concatenated with the noise latent $\boldsymbol{\epsilon} \in \mathbb{R}^{t\times 2h\times 2w\times c}$ along the channel dimension, yielding a $2c$-channel input. The DiT's input projection is extended from c to $2c$ channels, with the new weights initialized to zero so that training begins from the original T2V checkpoint. This zero-initialization preserves the model's generative capability at the start of training, and the model gradually learns to leverage the LR condition as training progresses. To further enhance robustness and preserve generative capacity, we apply two conditioning augmentation strategies during training: ***(i) Condition noise injection***: Gaussian noise at a random level $\sigma_{\text{cond}} \in [\sigma_{\min}, \sigma_{\max}]$ is added to the LR condition latent before concatenation, preventing the model from overly relying on the condition and encouraging it to leverage its learned generative priors to complement missing detail. ***(ii) Condition dropout***: with probability p_{drop}, the LR condition is entirely zeroed out, forcing the model to perform pure T2V generation without any visual condition. This ensures the SR model

retains strong unconditional generative capability, which both improves classifier-free guidance effectiveness and prevents mode collapse onto the LR input.

AIGC-Oriented Data Degradation Pipeline. Standard degradation models designed for natural video He et al. (2024) (*e.g.*, Real-ESRGAN Wang et al. (2021)) are poorly suited for AI-generated content, which exhibits characteristic artifacts distinct from natural camera noise—temporal flickering, unnatural motion jitter, rolling-shutter wobble, and diffusion-specific color shifts. We design a hybrid two-stage degradation pipeline that combines AIGC-specific temporal degradation with classical spatial degradation:

Stage 1—AIGC synthetic degradation. A dedicated module applies five temporally coherent operations whose parameters evolve smoothly over time via low-frequency sinusoidal trajectories (avoiding inter-frame flicker): *(a) Temporal morphing*: adjacent frames are alpha-blended with a time-varying mixing coefficient $\alpha_t \in [0.2, 0.9]$, simulating the exposure fusion artifacts common in diffusion-generated video. *(b) Stochastic frame dropping*: frames are randomly dropped (probability p_{drop} with a maximum consecutive-drop constraint) and reconstructed via linear interpolation, emulating temporal jitter from autoregressive generation. *(c) Directional motion blur*: per-frame line kernels with temporally smooth angle θ_t and length l_t produce spatially varying motion blur, mimicking the anisotropic blur patterns unique to diffusion denoising. *(d) ROI-constrained grid warping*: a low-frequency displacement field is generated and masked by a temporally drifting soft-ellipse ROI, producing localized geometric distortion that resembles rolling-shutter wobble in AI-generated videos. *(e) Video codec compression*: H.264 encoding at randomized CRF levels introduces block artifacts and quantization noise characteristic of compressed AIGC outputs.

Stage 2—Cascaded spatial degradation. Following the AIGC stage, a Real-ESRGAN-style Wang et al. (2021) two-pass spatial degradation is applied: each pass consists of USM sharpening, Gaussian blur (kernel size $\in [15, 37]$, $\sigma \in [0.2, 3.0]$), random rescaling (factor $\in [0.15, 1.5]$), additive noise, and JPEG compression ($q \in [70, 95]$). The two stages are cascaded to produce diverse, realistic degradation. Finally, a $2\times$–$4\times$ bicubic downsampling followed by upsampling back to the original resolution simulates the spatial resolution gap. A stochastic mixing strategy (CutMix between the AIGC-degraded and spatially-degraded branches) further enriches training diversity.

This hybrid pipeline generates realistic LR–HR training pairs that faithfully simulate the degradation characteristics of AI-generated video, enabling the SR model to effectively restore AIGC-specific artifacts while retaining the base model's generative priors.

SR Training Objective. The SR model is trained with the standard flow matching objective Liu et al. (2023). Given the clean HR latent $\mathbf{z}_0$ and sampled noise $\boldsymbol{\epsilon}$, we construct the noisy latent $\mathbf{z}_t = (1 - \sigma_t)\mathbf{z}_0 + \sigma_t\boldsymbol{\epsilon}$ at timestep t, where σ_t is drawn from a log-normal distribution with a shifted schedule (flow shift s). The model f_θ predicts the velocity field, trained via:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\boldsymbol{\epsilon}} \left\| f_\theta(\mathbf{z}_t,\, t,\, \mathbf{c}_{\text{text}},\, \mathbf{z}^{\text{HR}}) - (\boldsymbol{\epsilon} - \mathbf{z}_0) \right\|_2^2, \tag{1}$$

where the conditioning $\mathbf{z}^{\text{HR}}$ (upsampled LR latent) is concatenated along the channel dimension and the text prompt $\mathbf{c}_{\text{text}}$ provides semantic guidance. Combined with the condition noise injection and dropout described above, this training scheme enables classifier-free guidance (CFG) at inference, where the model can be steered between conditional SR and unconditional generation. At inference, multi-step ODE integration with CFG yields high-quality HR outputs from the trained flow. This multi-step SR model serves as the teacher for the subsequent single-step distillation stage (§2.3).

2.2 Causal Streaming Latent Upsampler with High-Resolution Decoder

In cascaded high-resolution (HR) generation, the upsampling stage bridges low-resolution (LR) latents and the subsequent SR model. As discussed in §1, existing approaches either use naive interpolation—introducing frequency aliasing that forces the SR stage to inject heavy noise—or employ heavyweight upsampler models that preclude streaming. We propose a unified causal memory network architecture that serves as both the streaming latent upsampler and the high-resolution decoder, achieving configurable spatiotemporal upsampling and decoding with minimal overhead.

Unified Causal Memory Network Architecture. Both the latent upsampler and the HR decoder share the same multi-stage architecture, differing only in their configured spatial/temporal scale factors and channel dimensions. An input 3×3 convolution followed by three cascaded stages, each consisting of N_b causal memory blocks, a spatial upsampling layer, a temporal expansion layer, and a

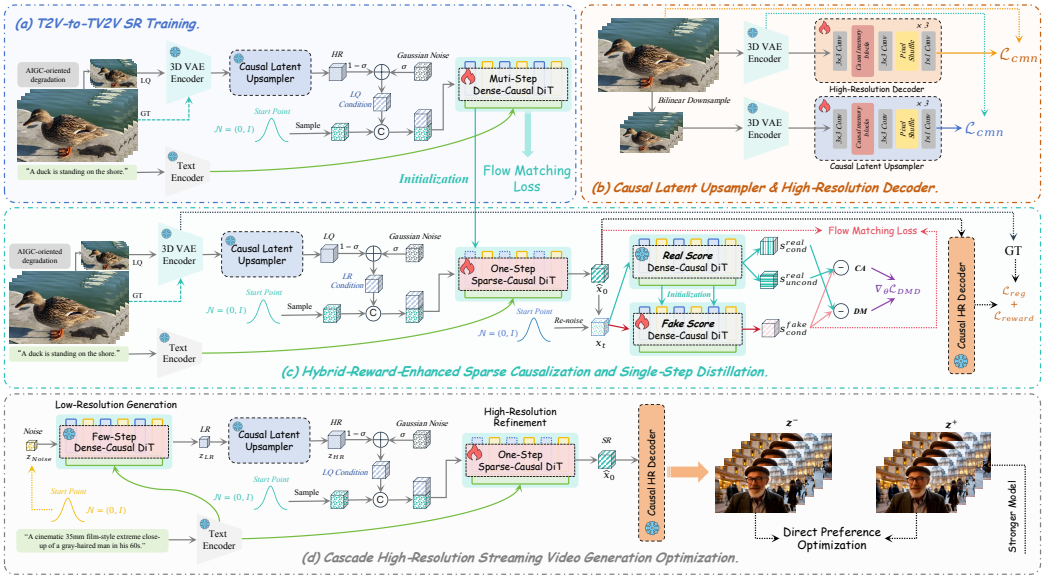


Figure 2: *Detailed components and training of our Ultra Flash framework. (Zoom in for details.)*

channel transition convolution. The core building block is the *CausalMemBlock*, which fuses the current frame's feature with the memory from the previous frame:

$$\mathbf{h}_t^{(\ell)} = \sigma\Big(\mathrm{Conv}_{3\times3}^{(3)}\Big(\mathrm{Conv}_{3\times3}^{(2)}\Big(\sigma\Big(\mathrm{Conv}_{3\times3}^{(1)}\Big([\mathbf{h}_t^{(\ell-1)};\ \mathbf{m}_{t-1}^{(\ell)}]\Big)\Big)\Big)\Big) + W_{\text{skip}}\mathbf{h}_t^{(\ell-1)}\Big), \tag{2}$$

where $[\cdot;\cdot]$ denotes channel concatenation, $\mathbf{m}_{t-1}^{(\ell)}$ is the previous frame's feature serving as temporal memory, σ is ReLU activation, and W_{skip} is a 1×1 projection for residual connection. The causal structure ensures each frame depends only on past context, enabling frame-wise streaming inference.

Spatial upsampling within each stage is performed via PixelShuffle Shi et al. (2016)—a 3×3 convolution expands the channel dimension by r^2 followed by a sub-pixel rearrangement that converts channels into spatial resolution (per-stage factor of $1\times$ or $2\times$ or $3\times$). Temporal upsampling is realized via a *temporal expansion* operator—a 1×1 convolution that lifts the channel dimension by factor s, followed by a channel-to-time reshape that unfolds s new frames from each input frame. By configuring stage factors independently, the same architecture supports $n\times$ spatial upsampling for the latent upsampler ($n = a \times b \times c$, spatial_factors$= [a, b, c]$, temporal_factors$= [1, 1, 1]$) and $8\times8\times4$ spatiotemporal decoding for the HR decoder (spatial_factors$= [2, 2, 2]$, temporal_factors$= [1, 2, 2]$).

Optimization. Both streaming latent upsampler and HR decoder are trained with an MSE reconstruction loss combined with an optical-flow-warped temporal consistency (eWarp) loss:

$$\mathcal{L}_{\text{cmn}} = \|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|_2^2 + \lambda_{\text{warp}} \cdot \mathcal{L}_{\text{eWarp}}, \tag{3}$$

where $\hat{\mathbf{x}}$ denotes the network output—latent features $\hat{\mathbf{z}}$ for the upsampler or decoded pixels $\hat{\mathbf{I}}$ for the HR decoder—and $\mathcal{L}_{\text{eWarp}} = \sum_t \|\hat{\mathbf{x}}_t - \text{Warp}(\hat{\mathbf{x}}_{t-1}, \mathbf{F}_{t\to t-1})\|$ penalizes temporal inconsistency by warping adjacent frames using the estimated optical flow $\mathbf{F}$. This encourages spatiotemporally smooth outputs, directly reducing the noise level required by the downstream SR model.

2.3 Cascaded High-Resolution Streaming Generation Optimization.

The multi-step SR model from §2.1 produces high-quality results but relies on dense bidirectional attention with iterative denoising, far from real-time. We devise a two-phase optimization scheme that progressively transforms it into a real-time streaming model: ***Phase I*** converts the SR model into a single-step causal generator via hybrid-reward-enhanced sparse causalization and distillation; ***Phase II*** closes the train-test gap of the full cascaded pipeline via self-forcing preference optimization, coupled with dynamic cache management that further improves inference efficiency.

2.3.1 Hybrid-Reward-Enhanced Sparse Causalization and Single-Step Distillation

This phase addresses two orthogonal bottlenecks simultaneously—the dense bidirectional attention precludes streaming, and multi-step denoising dominates latency—while injecting perceptual reward signals to compensate for the quality degradation typically incurred by acceleration.

Dynamic Block-Sparse Causal Attention. To enable streaming-compatible inference, we replace the dense bidirectional attention with *dynamic block-sparse causal attention*. The 3D token grid (t, h, w) is divided into non-overlapping blocks of (b_t, b_h, b_w), yielding N_b blocks. For each attention layer, a block-level mask is computed in two stages: ***(a) Structural masks:*** a spatial locality mask $\mathbf{M}_{\text{local}} \in \{0,1\}^{N_b \times N_b}$ confines each block's receptive field to a sliding window of size $r \times r$, and a temporal causal mask $\mathbf{M}_{\text{causal}}$ ensures each temporal chunk attends only to current and preceding chunks. ***(b) Content-adaptive top-k selection:*** we pool Q and K within each block, compute block-level attention scores, and retain the top-k most relevant blocks per head:

$$s_{ij}^h = \frac{\bar{\mathbf{q}}_i^{h\top} \bar{\mathbf{k}}_j^h}{\sqrt{d_h}}, \quad \mathbf{M}_{\text{sparse}}^h[i,j] = \mathbb{1}\big[\text{softmax}(s_{i,:}^h)_j \geq \tau_k\big] \cap \mathbf{M}_{\text{local}} \cap \mathbf{M}_{\text{causal}}, \tag{4}$$

where h indexes the attention head, $\bar{\mathbf{q}}_i, \bar{\mathbf{k}}_j$ are block-mean pooled queries and keys, d_h is the per-head dimension, τ_k is the adaptive threshold corresponding to the top-k budget, and $\mathbb{1}[\cdot]$ is the indicator function. The per-head mask is passed to a block-sparse attention kernel Guo et al. (2024) for hardware-efficient execution, with the top-k budget scaling adaptively with resolution to maintain consistent sparsity.

Single-Step Distillation via Decoupled DMD. We adopt Decoupled DMD Liu et al. (2025) to compress multi-step denoising into a single forward pass. The distillation involves three models: *(i)* the *real score model* (teacher)—the multi-step bidirectional SR model with CFG trained in §2.1, kept frozen; *(ii)* the *fake score model*—initialized from the teacher weights and trained on student-generated samples via a flow matching objective to track the evolving generator distribution, updated at 5× the generator's frequency; and *(iii)* the *generator* (student)—converted to causal sparse attention and being the primary trainable component. The decoupled DMD gradient decomposes into a CFG Augmentation (CA) term that drives the multi-step-to-single-step conversion, and a Distribution Matching (DM) regularizer that stabilizes generation quality:

$$\nabla_\theta \mathcal{L}_{\text{d-DMD}} = \mathbb{E}\left[- \left(\underbrace{s_{\text{cond}}^{\text{real}}(\mathbf{x}_{\tau_{\text{DM}}}) - s_{\text{cond}}^{\text{fake}}(\mathbf{x}_{\tau_{\text{DM}}})}_{\text{DM regularizer}} + (\alpha - 1) \underbrace{\big(s_{\text{cond}}^{\text{real}}(\mathbf{x}_{\tau_{\text{CA}}}) - s_{\text{uncond}}^{\text{real}}(\mathbf{x}_{\tau_{\text{CA}}})\big)}_{\text{CA engine}} \right) \frac{\partial G_\theta}{\partial \theta} \right], \tag{5}$$

where $s_{\text{cond/uncond}}^{\text{real}}$ and $s_{\text{cond}}^{\text{fake}}$ denote the conditional/unconditional score predictions from the real and fake models respectively, α is the CFG scale, G_θ is the student generator, and $\tau_{\text{CA}} > t$, $\tau_{\text{DM}} \in [0,1]$ are decoupled re-noising schedules. Since the SR task has access to ground-truth HR targets, we further introduce a wavelet L1 loss (omitting the LL sub-band to emphasize high-frequency detail) and an LPIPS perceptual loss to constrain pixel-level reconstruction via the HR decoder:

$$\mathcal{L}_{\text{reg}} = \lambda_{\text{wav}} \left\| \mathcal{W}_{\text{HF}}(\mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_0)) - \mathcal{W}_{\text{HF}}(\mathbf{I}_{\text{gt}}) \right\|_1 + \lambda_{\text{lpips}} \cdot \text{LPIPS}(\mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_0), \mathbf{I}_{\text{gt}}), \tag{6}$$

where $\mathcal{W}_{\text{HF}}$ denotes the high-frequency wavelet sub-bands (LH, HL, HH) and $\mathcal{D}_{\text{HR}}$ is the differentiable HR decoder enabling gradient back-propagation to the student.

Hybrid Reward Integration. Beyond reconstruction losses, we integrate perceptual and aesthetic reward signals from frozen quality predictors to directly optimize the student's visual quality:

$$\mathcal{L}_{\text{reward}} = -\lambda_{\text{clip}} \cdot \text{CLIP-IQA}^+(\mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_0)) - \lambda_{\text{musiq}} \cdot \text{MUSIQ}(\mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_0)) - \lambda_{\text{aes}} \cdot \text{LAION-Aes}(\mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_0)), \tag{7}$$

where CLIP-IQA$^+$ Wang et al. (2023a) captures perceptual quality, MUSIQ Ke et al. (2021) evaluates multi-scale image quality, and the LAION-Aesthetic predictor Schuhmann et al. (2022) assesses aesthetic appeal. Gradients flow through $\mathcal{D}_{\text{HR}}$ back to the student, providing complementary signals that directly enhance sharpness, color fidelity, and visual aesthetics beyond what distribution matching alone achieves. The SR model of student's total training objective in Phase I is:

$$\mathcal{L}_{\text{Phase I}} = \mathcal{L}_{\text{d-DMD}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{reward}}. \tag{8}$$

Table 1: **VBench comparison.** SC: subject consistency; BC: background consistency; MS: motion smoothness; IQ: imaging quality; AQ: aesthetic quality. ↑: higher is better.

Method	Steps	SC ↑	BC ↑	MS ↑	IQ ↑	AQ ↑	Total ↑
Wan2.1	50	96.87	97.35	98.31	65.02	62.79	83.37
CausVid Yin et al. (2025)	4	96.58	96.74	97.82	67.45	63.87	81.33
Self Forcing Huang et al. (2025)	4	97.12	97.08	97.63	67.92	63.50	83.74
Causal Forcing Zhu et al. (2026)	4	97.53	97.19	98.05	68.88	64.15	84.04
DummyForcing Guo et al. (2026)	4	96.45	96.81	98.14	67.28	63.62	83.48
Ultra Flash (Ours)	4	97.68	97.42	98.13	68.90	64.21	84.17

2.3.2 Cascaded Streaming Self-Forcing Preference Optimization and Cache Management

Phase I produces a single-step causal SR model, but it is still trained on ground-truth low-resolution context. At inference, the SR model must instead operate on imperfect outputs from the upstream streaming generator and its own prior predictions—a compounded exposure bias unique to the cascaded setting. Phase II addresses this via joint cascaded rollout with preference optimization.

High-Resolution Self-Forcing Rollout. During training, we simulate the actual inference distribution by performing high-resolution rollout of the entire cascaded pipeline: the low-resolution streaming generator produces context chunks autoregressively, which are upsampled by the latent upsampler and fed to the single-step SR model. The SR output in turn serves as context for subsequent chunks, exposing the model to its own imperfections and upstream errors simultaneously. By adjusting the spatial upsampling factor of the latent upsampler, we flexibly support both 1K and 2K streaming generation.

Preference Optimization. We apply Direct Preference Optimization (DPO) Rafailov et al. (2023) to the entire cascaded streaming pipeline, updating only the SR model's parameters. Preference pairs are constructed as follows: *negative samples* $\mathbf{z}^-$ are generated by the current cascaded pipeline in streaming mode (single-step causal SR), while *positive samples* $\mathbf{z}^+$ are produced by a stronger Wan2.2-5B SR model performing multi-step pixel-space super-resolution, serving as an oracle reference. The DPO loss directly optimizes the SR model to shift its output distribution toward the higher-quality reference:

$$\mathcal{L}_{\text{Phase II}} = \mathcal{L}_{\text{pref}} = -\log \sigma\left(\beta \left(\log \frac{\pi_\theta(\mathbf{z}^+|\mathbf{c})}{\pi_{\text{ref}}(\mathbf{z}^+|\mathbf{c})} - \log \frac{\pi_\theta(\mathbf{z}^-|\mathbf{c})}{\pi_{\text{ref}}(\mathbf{z}^-|\mathbf{c})}\right)\right), \tag{9}$$

where π_θ is the SR model being optimized, π_{ref} is the frozen reference policy (the Phase I checkpoint), $\mathbf{c}$ denotes the conditioning context from the upstream cascaded pipeline, and β is the temperature.

Dynamic Cache Management. At inference, we apply three complementary strategies to reduce per-chunk computation across the cascaded pipeline. ***(i) LR denoising step reduction.*** The upstream LR streaming generator (*e.g.*, Self Forcing Huang et al. (2025)) nominally uses 4 denoising steps per chunk. Since the downstream SR model compensates for fine detail, the LR output does not require full fidelity. We therefore run the complete 4-step schedule only for the first chunk and reduce all subsequent chunks to 3 steps, saving one forward pass per chunk without perceptible quality degradation. ***(ii) Adaptive cache refresh.*** By default, after the denoising steps the generator performs an additional forward pass on the predicted $\mathbf{x}_0$ to compute fresh KV entries for the cache. We instead evaluate the previous chunk's decoded frames with a lightweight IQA metric: if the score exceeds a predefined threshold, we directly reuse the KV cache from the last denoising step $\mathbf{x}_t$, eliminating the extra forward pass and further reducing latency. ***(iii) SR cache length adaptation.*** For the single-step SR model, the strong conditioning signal from the upsampled LR latent makes generation quality relatively insensitive to KV cache length. We exploit this by dynamically selecting a compact cache window, trading minimal quality for significant memory and compute savings during extended sequence generation.

Table 2: **Efficiency comparison.** Throughput (FPS), per-frame latency, and peak GPU memory for streaming generation. All measurements on a single GPU.

Method	Resolution	Steps	FPS ↑	Latency (ms) ↓	Streaming
Wan2.1 Team Wan et al. (2025)	480×832	50	0.78	103,000	×
CausVid Yin et al. (2025)	480×832	4	29.4	34	✓
Self Forcing Huang et al. (2025)	480×832	4	32.0	31	✓
Causal Forcing Zhu et al. (2026)	480×832	4	31.2	32	✓
DummyForcing Guo et al. (2026)	480×832	4	28.0	36	✓
Self Forcing + FlashVSR Zhuang et al. (2025)	768×1408	5	15.0	67	✓
Ultra Flash (Ours)	960×1664 (1K)	4	30.0	40	✓
Ultra Flash (Ours)	1440×2496 (2K)	4	18.0	56	✓

Table 3: **SR quality comparison.** We compare upsampling strategies (interpolation *vs.* streaming latent upsampler) and optimization stages (multi-step *vs.* distilled *vs.* full pipeline).

Configuration	CLIP-IQA+ ↑	MUSIQ ↑	NIQE ↓	FPS ↑
Bilinear + Multi-step SR	0.612	68.3	4.82	1.8
Bicubic + Multi-step SR	0.625	69.1	4.65	1.8
Upsampler + Multi-step SR	0.671	72.4	4.18	1.6
Upsampler + Sparse single-step	0.658	71.2	4.31	30.0
Full pipeline (+ pref. opt.)	0.692	73.8	3.95	30.0

3 Experiments

3.1 Experimental Setup

Base Model and Pipeline. We build Ultra Flash on top of Wan2.1-T2V-1.3B Team Wan et al. (2025), a 1.3B-parameter video DiT with 30 transformer blocks, hidden dimension 1536, 12 attention heads, and a UMT5-XXL text encoder. A pre-trained low-resolution streaming generator produces 480P latents, which are fed to the streaming latent upsampler, the generative SR model, and the HR decoder in cascade.

Evaluation. We evaluate on **VBench** Huang et al. (2024), perceptual quality metrics (CLIP-IQA+, MUSIQ, NIQE), and efficiency metrics (FPS, latency per chunk) on a single H200/B200 GPU.

Baselines. We compare with: (1) **CausVid** Yin et al. (2025): autoregressive DMD with dense attention; (2) **Self Forcing** Huang et al. (2025): self-rollout training for AR video; (3) **Causal Forcing** Zhu et al. (2026): improved AR distillation; (4) **DummyForcing** Guo et al. (2026): dummy-head acceleration; (5) **FlashVSR** Zhuang et al. (2025): sparse-attention pixel-space streaming SR. All streaming baselines operate at 480P; FlashVSR operates at 768×1408 with an external LQ video input. ***More details can be found in the appendix.***

3.2 Main Results

Quality Comparison. Table 1 reports VBench scores. Ultra Flash achieves competitive or superior quality to prior few-step methods on all dimensions while being the only method that additionally supports real-time high-resolution generation.

Efficiency and Resolution Scaling. Table 2 compares efficiency across methods and resolutions. Ultra Flash achieves ∼30 FPS at 1K (960×1664) and ∼18 FPS at 2K (1440×2496) on a single B200 GPU and ∼17 FPS & ∼10 FPS on a single H200 GPU, significantly outperforming existing methods.

High-Resolution SR Quality. Table 3 evaluates the generative SR quality at 1K resolution, comparing different upsampling strategies and optimization stages within the Ultra Flash pipeline.

3.3 Ablation Studies

We ablate the three core contributions of Ultra Flash to validate each component. All ablations are conducted at 960×1664 unless stated otherwise.

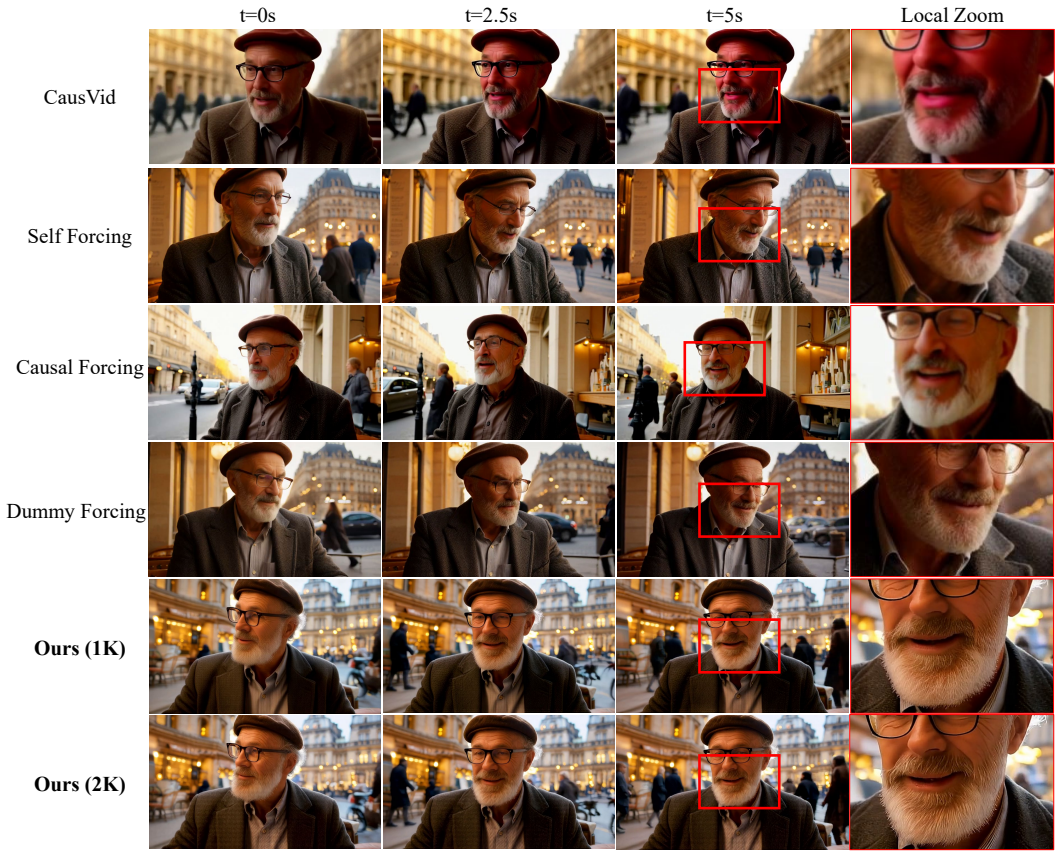


Figure 3: **Qualitative comparison.** Ultra Flash real-time generates sharper, temporally coherent frames at 1K & 2K while prior methods are limited to 480×832. ***Zoom in for the details.***

Table 4: **Ablation: SR training components.** Impact of AIGC degradation, zero-init conditioning, condition noise injection, and condition dropout.

Configuration	CLIP-IQA+ ↑	MUSIQ ↑	NIQE ↓	LPIPS ↓
Full SR training	0.671	72.4	4.18	0.182
− AIGC degradation	0.638	69.5	4.67	0.205
− Zero-init conditioning	0.621	68.1	4.83	0.218
− Condition noise injection	0.644	70.2	4.52	0.197
− Condition dropout	0.652	70.8	4.41	0.193

Table 5: **Ablation: upsampling strategy.** Impact on downstream SR quality and latency overhead.

Upsampler	CLIP-IQA+ ↑	MUSIQ ↑	NIQE ↓	Overhead (ms)
Bilinear interpolation	0.612	68.3	4.82	<1
Bicubic interpolation	0.625	69.1	4.65	<1
Causal memory (4 blocks)	0.658	71.6	4.28	5
Causal memory (8 blocks)	0.671	72.4	4.18	8

Architecture-Preserving SR Training. Table 4 validates the SR training paradigm. Removing the AIGC-oriented degradation pipeline (using standard Real-ESRGAN degradation only) reduces quality, confirming that tailored degradation effectively simulates AI-generated artifacts. Removing zero-initialized conditioning (replacing with random init) destabilizes early training. Disabling condition noise injection causes the model to overfit to the LR input and lose generative capability, while disabling condition dropout weakens CFG effectiveness.

Table 6: **Ablation: cascaded streaming optimization components.** Each row removes one component from the full pipeline. TC: temporal consistency (VBench). Mem: peak GPU memory.

Configuration	CLIP-IQA+ ↑	MUSIQ ↑	TC ↑	FPS ↑	Mem (GB) ↓
Full pipeline	0.692	73.8	97.2	30.0	12.6
− Adaptive top-k (fixed window)	0.674	72.1	96.8	28.5	12.6
− Hybrid reward $\mathcal{L}_{\text{reward}}$	0.665	71.4	97.0	30.0	12.6
− DPO pref. $\mathcal{L}_{\text{pref}}$	0.673	72.5	95.6	30.0	12.6
− Dynamic cache management	0.690	73.6	97.1	27.4	16.8

Causal Streaming Latent Upsampler. Table 5 compares upsampling strategies. Naive interpolation introduces frequency aliasing that degrades downstream SR quality (lower CLIP-IQA+ and MUSIQ), while the causal memory network produces spatiotemporally coherent latents that substantially ease the SR task. Increasing the number of memory blocks from 4 to 8 further improves quality with marginal overhead ($<5\%$ of total pipeline latency).

Cascaded Streaming Optimization. We ablate each component of the streaming optimization scheme in Table 6. ***(i) Sparse attention:*** Dense causal attention triggers out-of-memory at 960×1664; block-sparse attention resolves this. Content-adaptive top-k selection (vs. fixed window) yields better quality by allowing global information routing where needed. ***(ii) Hybrid reward:*** Adding CLIP-IQA$^+$, MUSIQ, and LAION-Aesthetic reward signals during distillation improves perceptual quality beyond what Decoupled DMD + reconstruction losses alone achieve. ***(iii) DPO preference optimization:*** The cascaded DPO loss with a stronger Wan2.2-5B teacher closes the train-test gap, improving temporal consistency and visual quality over long streaming sequences. ***(iv) Dynamic cache management:*** The three-pronged inference optimization (step reduction, IQA-adaptive cache refresh, SR cache length adaptation) significantly improves FPS with negligible quality impact.

3.4 Qualitative Results

Figure 3 presents a visual comparison; Appendix D includes additional comparisons in Figures 4 and 5. Prior streaming methods generate at 480×832 with visible artifacts (blurriness, flickering), while Ultra Flash produces sharp, temporally coherent frames at 1K & 2K with rich high-frequency detail. The causal streaming latent upsampler avoids aliasing artifacts visible in naive-interpolation baselines, the hybrid reward signals yield sharper textures and more natural colors, and the cascaded DPO preserves quality over extended sequences.

4 Conclusion

We presented Ultra Flash, a cascaded streaming framework that scales real-time autoregressive video generation from 480P to 1K ($\sim$30 FPS) and 2K ($\sim$18 FPS) on a single GPU. The key contributions—architecture-preserving SR training with AIGC degradation, a causal streaming latent upsampler with HR decoder, and cascaded streaming optimization via sparse distillation and DPO—jointly advance the resolution–speed Pareto frontier while maintaining state-of-the-art visual quality. Limitations and broader impact are discussed in Appendix B.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Ollin Boer Bohan. TAEHV: Tiny autoencoder for hunyuan video. `https://github.com/madebyollin/taehv`, 2025.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI Technical Report, 2024. `https://openai.com/index/video-generation-models-as-world-simulators/`.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiaxiang Cheng, Pan Xie, Xin Xia, Jiashi Li, Jie Wu, Yuxi Ren, Huixia Li, Xuefeng Xiao, Min Zheng, and Lean Fu. Resadapter: Domain consistent resolution adapter for diffusion models. *arXiv preprint arXiv:2403.02084*, 2024.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

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. *arXiv preprint arXiv:2510.02283*, 2025.

Ruoyi Du, Dongliang Chang, Timothy Hospedales, Yi-Zhe Song, and Zhanyu Ma. Demofusion: Democratising high-resolution image generation with no $$$. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Zihao Fan, Xin Lu, Yidi Liu, Jie Huang, Dong Li, Xueyang Fu, and Baocai Yin. Bird-sr: Bidirectional reward-guided diffusion for real-world image super-resolution. *arXiv preprint arXiv:2602.07069*, 2026.

FSVideo Team, Qingyu Chen, Zhiyuan Fang, Haibin Huang, Xinwei Huang, Tong Jin, Minxuan Lin, Bo Liu, Celong Liu, Chongyang Ma, Xing Mei, Xiaohui Shen, Yaojie Shen, Fuwen Tan, Angtian Wang, Xiao Yang, Yiding Yang, Jiamin Yuan, Lingxi Zhang, and Yuxin Zhang. Fsvideo: Fast speed video diffusion model in a highly-compressed latent space. *arXiv preprint arXiv:2602.02092*, 2026.

Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, Fangyuan Kong, Huixia Li, Jiashi Li, Liang Li, Xiaojie Li, Xunsong Li, Yifu Li, Shanchuan Lin, Zhijie Lin, Jiawei Liu, Shu Liu, Xiaonan Nie, Zhiwu Qing, Yuxi Ren, Li Sun, Zhi Tian, Rui Wang, Sen Wang, Guoqiang Wei, Guohong Wu, Jie Wu, Ruiqi Xia, Fei Xiao, et al. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025.

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

Junxian Guo, Haotian Tang, Shang Yang, Zhekai Zhang, Zhijian Liu, and Song Han. Block Sparse Attention. `https://github.com/mit-han-lab/Block-Sparse-Attention`, 2024.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. LTX-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

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, et al. LTX-2: Efficient joint audio-visual foundation model. *arXiv preprint arXiv:2601.03233*, 2026.

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.

Yingqing He, Shaoshu Yang, Haoxin Chen, Xiaodong Cun, Menghan Xia, Yong Zhang, Xintao Wang, Ran He, Qifeng Chen, and Ying Shan. Scalecrafter: Tuning-free higher-resolution visual generation with diffusion models. *arXiv preprint arXiv:2310.07702*, 2023.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022.

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

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 (CVPR)*, 2024.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. MUSIQ: Multi-scale image quality transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Jihwan Kim, Junoh Kang, Jinyoung Choi, and Bohyung Han. FIFO-diffusion: Generating infinite videos from text without training. *arXiv preprint arXiv:2405.11473*, 2024.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*, 2020.

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, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Dongyang Liu, Peng Gao, David Liu, Ruoyi Du, Zhen Li, Qilong Wu, Xin Jin, Sihan Cao, Shifeng Zhang, Hongsheng Li, and Steven Hoi. Decoupled dmd: Cfg augmentation as the spear, distribution matching as the shield. *arXiv preprint arXiv:2511.22677*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Xin Lu, Yurui Zhu, Xi Wang, Dong Li, Jie Xiao, Yunpeng Zhang, Xueyang Fu, and Zheng-Jun Zha. Hirformer: Dynamic high resolution transformer for large-scale image shadow removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 6513–6523, June 2024.

Xin Lu, Yuanfei Bao, Jiarong Yang, Anya Hu, Jie Xiao, Kunyu Wang, Dong Li, Senyan Xu, Kean Liu, Xueyang Fu, and Zheng-Jun Zha. Evenformer: Dynamic even transformer for real-world image restoration. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR) Workshops*, pp. 1081–1091, June 2025a.

Xin Lu, Xueyang Fu, Jie Xiao, Zihao Fan, Yurui Zhu, and Zheng-Jun Zha. Elucidating and endowing the diffusion training paradigm for general image restoration. *arXiv preprint arXiv:2506.21722*, 2025b.

Xin Lu, Yufeng Peng, Chengjie Ge, Zhijing Sun, Ziang Zhou, Zishun Liao, Zihao Li, Dong Li, Qiyu Kang, Xueyang Fu, and Zheng-Jun Zha. Boosting inverse tone mapping via diffusion regularization. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 5654–5663, 2025c. doi: 10.1109/ICCVW69036.2025.00591.

Xin Lu, Zhijing Sun, Chengjie Ge, Yufeng Peng, Ziang Zhou, Zihao Li, Zishun Liao, Dong Li, Qiyu Kang, Xueyang Fu, and Zheng-Jun Zha. Efficient high fps non-uniform motion deblurring via progressive learning. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 5644–5653, 2025d. doi: 10.1109/ICCVW69036.2025.00590.

Xin Lu, Jie Xiao, Yurui Zhu, and Xueyang Fu. Continuous adverse weather removal via degradation-aware distillation. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 28113–28123, June 2025e.

Xin Lu, Jiarong Yang, Yuanfei Bao, Zihao Fan, Anya Hu, Kunyu Wang, Jie Xiao, Xi Wang, Hongjian Liu, Xueyang Fu, and Zheng-Jun Zha. Advancing ambient lighting normalization via diffusion shadow generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR) Workshops*, pp. 1070–1080, June 2025f.

Xin Lu, Zihao Fan, Jie Huang, Mingchen Zhong, Hexin Zhang, Xueyang Fu, and Zheng-Jun Zha. Omnivr: Audio-video conditional generation for archival footage restoration. *arXiv preprint arXiv:2608.04224*, 2026a.

Xin Lu, Jie Huang, Jie Xiao, Dong Li, and Xueyang Fu. Expert weaving: Marrying masked auto-regressive and diffusion models for unified image restoration. In *European Conference on Computer Vision*, pp. 173–193. Springer, 2026b.

Yunhong Lu, Yanhong Zeng, Haobo Li, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jiapeng Zhu, Hengyuan Cao, Zhipeng Zhang, Xing Zhu, Yujun Shen, and Min Zhang. Reward forcing: Efficient streaming video generation with rewarded distribution matching distillation. *arXiv preprint arXiv:2512.04678*, 2025g.

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, et al. Longcat-video technical report. *arXiv preprint arXiv:2510.22200*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Long Peng, Aiwen Jiang, Qiaosi Yi, and Mingwen Wang. Cumulative rain density sensing network for single image derain. *IEEE Signal Processing Letters*, 27:406–410, 2020.

Long Peng, Aiwen Jiang, Haoran Wei, Bo Liu, and Mingwen Wang. Ensemble single image deraining network via progressive structural boosting constraints. *Signal Processing: Image Communication*, 99:116460, 2021.

Long Peng, Yang Cao, Renjing Pei, Wenbo Li, Jiaming Guo, Xueyang Fu, Yang Wang, and Zheng-Jun Zha. Efficient real-world image super-resolution via adaptive directional gradient convolution. *arXiv preprint arXiv:2405.07023*, 2024a.

Long Peng, Wenbo Li, Jiaming Guo, Xin Di, Haoze Sun, Yong Li, Renjing Pei, Yang Wang, Yang Cao, and Zheng-Jun Zha. Unveiling hidden details: A raw data-enhanced paradigm for real-world super-resolution. *arXiv preprint arXiv:2411.10798*, 2024b.

Long Peng, Wenbo Li, Renjing Pei, Jingjing Ren, Jiaqi Xu, Yang Wang, Yang Cao, and Zheng-Jun Zha. Towards realistic data generation for real-world super-resolution. *arXiv preprint arXiv:2406.07255*, 2024c.

Long Peng, Wenbo Li, Jiaming Guo, Xin Di, Haoze Sun, Yong Li, Renjing Pei, Yang Wang, Yang Cao, and Zheng-Jun Zha. Boosting real-world super-resolution with raw data: a new perspective, dataset and baseline. OpenReview, 2025a. URL `https://openreview.net/forum?id=WxKS3XV7Z4`. ICLR 2025 submission.

Long Peng, Anran Wu, Wenbo Li, Peizhe Xia, Xueyuan Dai, Xinjie Zhang, Xin Di, Haoze Sun, Renjing Pei, Yang Wang, et al. Pixel to gaussian: Ultra-fast continuous super-resolution with 2d gaussian modeling. *arXiv preprint arXiv:2503.06617*, 2025b.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jingjing Ren, Wenbo Li, Haoyu Chen, Renjing Pei, Bin Shao, Yong Guo, Long Peng, Fenglong Song, and Lei Zhu. Ultrapixel: Advancing ultra high-resolution image synthesis to new peaks. *Advances in Neural Information Processing Systems*, 37:111131–111171, 2024.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. *arXiv preprint arXiv:2402.09470*, 2024.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883, 2016.

Hau-Shiang Shiu, Chin-Yang Lin, Zhixiang Wang, Chi-Wei Hsiao, Po-Fan Yu, Yu-Chih Chen, and Yu-Lun Liu. Stream-diffvsr: Low-latency streamable video super-resolution via auto-regressive diffusion. *arXiv preprint arXiv:2512.23709*, 2025.

SII-GAIR, Sand.ai, Ethan Chern, Hansi Teng, Hanwen Sun, Hao Wang, Hong Pan, Hongyu Jia, Jiadi Su, Jin Li, Junjie Yu, Lijie Liu, Lingzhi Li, Lyumanshan Ye, Min Hu, Qiangang Wang, Quanwei Qi, Steffi Chern, Tao Bu, Taoran Wang, Teren Xu, Tianning Zhang, Tiantian Mi, Weixian Xu, Wenqiang Zhang, Wentai Zhang, Xianping Yi, Xiaojie Cai, Xiaoyang Kang, Yan Ma, Yixiu Liu, Yunbo Zhang, Yunpeng Huang, Yutong Lin, Zewei Tao, Zhaoliang Liu, Zheng Zhang, Zhiyao Cen, Zhixuan Yu, Zhongshu Wang, Zhulin Hu, Zijin Zhou, Zinan Guo, Yue Cao, and Pengfei Liu. Speed by simplicity: A single-stream architecture for fast audio-video generative foundation model. *arXiv preprint arXiv:2603.21986*, 2026.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023.

Team 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, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2019.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*, 2024.

Jianyi Wang, Kelvin C.K. Chan, and Chen Change Loy. Exploring CLIP for assessing the look and feel of images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 2555–2563, 2023a.

Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. In *International Conference on Computer Vision Workshops (ICCVW)*, 2021.

Yang Wang, Long Peng, Liang Li, Yang Cao, and Zheng-Jun Zha. Decoupling-and-aggregating for image exposure correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18115–18124, 2023b.

Yingqian Wang, Zhengyu Liang, Fengyuan Zhang, Lvli Tian, Longguang Wang, Juncheng Li, Jungang Yang, Radu Timofte, Yulan Guo, Kai Jin, et al. Ntire 2025 challenge on light field image super-resolution: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1227–1246, 2025.

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, et al. Hunyuanvideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025.

Jie Xiao, Xueyang Fu, Aiping Liu, Feng Wu, and Zheng-Jun Zha. Image de-raining transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12978–12995, 2023. doi: 10.1109/TPAMI.2022.3183612.

Jie Xiao, Xueyang Fu, Yurui Zhu, and Zheng-Jun Zha. Bayesian window transformer for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(3):2431–2444, 2026. doi: 10.1109/TPAMI.2025.3626947.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhucun Xue, Jiangning Zhang, Teng Hu, Haoyang He, Yinan Chen, Yuxuan Cai, Yabiao Wang, Chengjie Wang, Yong Liu, Xiangtai Li, and Dacheng Tao. Ultravideo: High-quality uhd video dataset with comprehensive captions. *arXiv preprint arXiv:2506.13691*, 2025.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. *arXiv preprint arXiv:2405.14867*, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

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

Guohui Zhang, XiaoXiao Ma, Jie Huang, Hang Xu, Hu Yu, Siming Fu, Yuming Li, Zeyue Xue, Lin Song, Haoyang Huang, et al. Omninft: Modality-wise omni diffusion reinforcement for joint audio-video generation. *arXiv preprint arXiv:2605.12480*, 2026.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025a.

Shilong Zhang, Wenbo Li, Shoufa Chen, Chongjian Ge, Peize Sun, Yifu Zhang, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Ping Luo. Flashvideo: Flowing fidelity to detail for efficient high-resolution video generation. *arXiv preprint arXiv:2502.05179*, 2025b.

Yifu Zhang, Hao Yang, Yuqi Zhang, Yifei Hu, Fengda Zhu, Chuang Lin, Xiaofeng Mei, Yi Jiang, Bingyue Peng, and Zehuan Yuan. Waver: Wave your way to lifelike video generation. *arXiv preprint arXiv:2508.15761*, 2025c.

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

Junhao Zhuang, Shi Guo, Xin Cai, Xiaohui Li, Yihao Liu, Chun Yuan, and Tianfan Xue. Flashvsr: Towards real-time diffusion-based streaming video super-resolution. *arXiv preprint arXiv:2510.12747*, 2025.

A Related Work

Video Diffusion Models. Recent advances in video generation have achieved remarkable progress, enabling the synthesis of high-fidelity and temporally coherent videos directly from textual prompts Yang et al. (2024); Kong et al. (2024); Team Wan et al. (2025). Early methods extend text-to-image diffusion models by introducing temporal modules to capture frame dynamics, yet often fail to model holistic spatiotemporal dependencies Ho et al. (2022); Guo et al. (2023); Blattmann et al. (2023). With the emergence of the diffusion transformer (DiT) Peebles & Xie (2023), transformer-based architectures have become the dominant paradigm, jointly modeling spatial and temporal correlations through full attention mechanisms Yang et al. (2024); Kong et al. (2024); HaCohen et al. (2024); Team Wan et al. (2025). Modern text-to-video (T2V) models typically adopt a framework consisting of a 3D VAE for spatiotemporal compression and a DiT for latent-space denoising. Building on this foundation, recent works, including CogVideoX Yang et al. (2024), HunyuanVideo Kong et al. (2024), and Wan Team Wan et al. (2025), further scale up model size and data, demonstrating impressive video quality at unprecedented levels. However, their non-autoregressive, bidirectional attention structure prevents streaming and incurs high latency for interactive use.

Autoregressive Video Diffusion. Given the inherent temporal order of video data, it is natural to model video generation as an autoregressive process. While most video diffusion models rely on bidirectional dependencies Brooks et al. (2024); Yang et al. (2024); Polyak et al. (2024), autoregressive video diffusion has recently been explored under two paradigms. *Teacher Forcing* Valevski et al. (2024); Jin et al. (2024) trains models to denoise new frames conditioned on clean context frames, while *Diffusion Forcing* Chen et al. (2024); Ruhe et al. (2024); Kim et al. (2024) supports autoregressive sampling by conditioning on frames with varying noise levels. CausVid Yin et al. (2025) first adapted bidirectional DiTs to autoregressive generation with causal attention, using ODE-trajectory initialization and asymmetric distribution matching distillation (DMD) Yin et al. (2024b) to reduce denoising steps. Self Forcing Huang et al. (2025) addressed the critical issue of *exposure bias*—the train-test mismatch where models trained on ground-truth context must generate conditioned on their own imperfect outputs—by simulating autoregressive rollouts during training. Causal Forcing Zhu et al. (2026) improved upon both with better ODE initialization and causal consistency distillation. More recent works extend this direction: Self-Forcing++ Cui et al. (2025) scales to minute-length videos, Reward Forcing Lu et al. (2025g) integrates reward signals into streaming distillation, and DummyForcing Guo et al. (2026) exploits redundant attention heads for training-free acceleration. Despite these advances, all existing methods remain confined to low resolutions (480P) and none addresses the quadratic attention bottleneck that prohibits high-resolution streaming.

Ultra-High-Resolution Video Generation. Ultra-high-resolution (UHR) video generation remains a fundamental challenge, hindered by immense computational demands and the scalability constraints of current models Xue et al. (2025); Lu et al. (2026a). Existing research primarily follows three paradigms. *Training-free methods* extend pre-trained diffusion models to higher resolutions without retraining by modifying denoising processes or attention structures He et al. (2023); Du et al. (2024), achieving computational efficiency but often producing over-smoothed textures and lacking genuine high-frequency detail. *Fine-tuning strategies* adapt low-resolution generative models on high-resolution datasets Cheng et al. (2024); Ren et al. (2024), enhancing fidelity while preserving generative priors. *Cascaded methods* have recently emerged as a promising direction: FlashVideo Zhang et al. (2025b) and LongCat Meituan LongCat Team et al. (2025) adopt two-stage pipelines with low-resolution generation followed by high-resolution refinement; Seedance Gao et al. (2025) demonstrates that motion dynamics are more effectively learned at lower resolutions; HunyuanVideo 1.5 Wu et al. (2025) and FSVideo FSVideo Team et al. (2026) employ latent-space upsampling followed by a large SR model but with no streaming capability; DaVinci SII-GAIR et al. (2026) directly applies interpolation upsampling in latent space, introducing frequency aliasing that burdens the subsequent SR stage; LTX-2 HaCohen et al. (2026) introduces a latent upsampler for multi-scale generation, but its upsampler is heavyweight and sensitive to sequence length, making it unsuitable for arbitrary-length streaming. These cascaded approaches are not designed for causal streaming and typically require heavy noise injection to compensate for upsampling artifacts, increasing SR training difficulty. Our work addresses these limitations by performing latent-space upsampling with a spatiotemporally coherent causal memory network, enabling real-time high-resolution streaming within a unified cascaded pipeline.

Efficient Attention for Video. The quadratic cost of attention is a primary bottleneck for high-resolution video. Sparse attention patterns—including local windows Beltagy et al. (2020), strided patterns Child et al. (2019), and learned sparsity Kitaev et al. (2020)—have been extensively explored for language and images. For video, SpargeAttn Zhang et al. (2025a) and Block-Sparse Attention Guo et al. (2024) provide hardware-efficient sparse kernels that accelerate attention computation. FlashVSR Zhuang et al. (2025) was the first to apply block-sparse attention to video super-resolution, introducing locality-constrained sparse patterns and a progressive distillation pipeline from dense to sparse attention. DummyForcing Guo et al. (2026) observed that $\sim$25% of attention heads in autoregressive video DiTs are "dummy" (attending only to the current frame) and exploited this for inference-time acceleration. However, FlashVSR requires projecting pixel-space low-quality video into the DiT latent space for conditional super-resolution, while DummyForcing uses sparsity only at inference without training adaptation. In contrast, our work integrates dynamic block-sparse attention into a pure generative streaming pipeline without any pixel-space input dependency, and achieves higher sparse efficiency through content-adaptive mask prediction within causal autoregressive distillation.

Distillation for Fast Generation. Distribution matching distillation (DMD) Yin et al. (2024b,a) reduces multi-step diffusion to few-step generation by matching output distributions via a learned critic. Decoupled DMD Liu et al. (2025) further improves this paradigm by separating CFG augmentation from distribution matching, achieving better quality–speed trade-offs. Consistency distillation Song et al. (2023) and rectified flow Liu et al. (2023) offer alternative paths to fast sampling. In the video domain, CausVid Yin et al. (2025) and Causal Forcing Zhu et al. (2026) extend DMD to autoregressive video with asymmetric teacher–student training, but use dense attention throughout. Reward Forcing Lu et al. (2025g) incorporates reward signals into DMD for streaming video but remains at low resolution. We build upon Decoupled DMD and augment it with perceptual and aesthetic reward signals Xu et al. (2023); Ke et al. (2021), directly optimizing for perceptual quality rather than surrogate losses, while simultaneously training with sparse causal attention to enable single-step high-resolution streaming generation.

Video Super-Resolution. Traditional models Zhuang et al. (2025); He et al. (2024); Lu et al. (2026b, 2025c,d); Fan et al. (2026); Lu et al. (2025b, 2024, 2025f,a,e); Wang et al. (2025); Peng et al. (2020); Wang et al. (2023b); Peng et al. (2024c, 2021); Ren et al. (2024); Peng et al. (2024a, 2025b, 2024b, 2025a); Xiao et al. (2026, 2023) achieves high fidelity but is computationally heavy, often requiring multiple denoising steps per frame. FlashVSR Zhuang et al. (2025) significantly accelerated diffusion-based VSR through block-sparse causal attention and a tiny conditional decoder, achieving near real-time streaming at 768×1408. Stream-DiffVSR Shiu et al. (2025) further explored autoregressive causal conditioning for low-latency streaming SR. However, both methods fundamentally operate as conditional super-resolution models that project pixel-space low-quality frames into the DiT latent space, requiring explicit LQ video input and architectural modifications that forfeit the generative capability of the pre-trained T2V model. Moreover, as Waver Zhang et al. (2025c) demonstrates, composing pixel-space SR with latent-space generators introduces additional encode–decode overhead that limits end-to-end efficiency. In contrast, our framework is a pure generative streaming pipeline that produces high-resolution video directly from text via an architecture-preserving training paradigm with an AIGC-oriented degradation pipeline, preserving the base model's generative priors. The streaming latent upsampler performs resolution scaling entirely in latent space with a causal memory network, seamlessly integrating into the end-to-end cascaded streaming pipeline without any pixel-space dependency.

B Limitations and Broader Impact

Limitations. (1) The block-sparse attention kernel achieves optimal hardware utilization on H200/B200 GPUs; performance on consumer-grade GPUs remains less optimized. (2) The DPO positive samples rely on a stronger Wan2.2-5B model; exploring online reward-based preference optimization could eliminate this dependency. (3) The current framework achieves real-time streaming at up to 2K resolution; scaling to 4K real-time generation remains beyond reach due to the quadratic growth of attention cost and memory bandwidth constraints. Achieving 4K real-time streaming video generation is a primary future research direction, potentially requiring advances in sub-linear attention mechanisms, more aggressive model compression, and hardware-software co-design.

Broader Impact. Real-time high-resolution video generation has broad positive potential in creative industries, accessibility, education, and interactive media. We acknowledge risks associated with deepfakes and misinformation, and advocate for robust watermarking, content provenance tracking, and responsible deployment practices.

C Implementation Details

Training Protocol. Training proceeds in four stages: *(i) Architecture-preserving SR fine-tuning* (§2.1): the T2V model is converted to a TV2V SR model via zero-initialized channel extension and trained on AIGC-degraded data with the flow matching objective (Eq. 1), condition noise injection ($\sigma_{\text{cond}} \in [0.4, 0.6]$), and condition dropout (p_{drop}=0.4). *(ii) Causal memory network pre-training* (§2.2): the streaming latent upsampler and HR decoder are trained on paired low-/high-resolution data with latent MSE and eWarp temporal consistency losses (Eq. 3). *(iii) Hybrid-reward-enhanced sparse causalization and single-step distillation* (Phase I, §2.3.1): the SR model is distilled via Decoupled DMD Liu et al. (2025) with block-sparse causal attention (block size $(2, 8, 8)$, adaptive top-k, local window 9×9), wavelet L1 + LPIPS reconstruction losses, and hybrid reward signals (CLIP-IQA$^+$, MUSIQ, LAION-Aesthetic) at high resolution (960×1664). *(iv) Cascaded streaming DPO* (Phase II, §2.3.2): the full cascaded pipeline is jointly rolled out, and a DPO loss (Eq. 9) aligns the SR model's output with a stronger Wan2.2-5B model. All stages use 32 GPUs (4 nodes $\times$ 8 H200), AdamW (lr=10^{-5}, β=$(0.9, 0.95)$), gradient clipping at 1.0, and bf16 mixed precision.

SR DiT Architecture. The SR model follows the Wan2.1 architecture with a $2c$-channel input (16 noise + 16 condition latent, zero-initialized extension): 30 transformer blocks, hidden dim 1536, FFN dim 8960, 12 heads (head dim 128); patch size $(1, 2, 2)$; 3D RoPE with axis dims $(44, 42, 42)$; sparse block size $(2, 8, 8)$; adaptive top-k with ρ=1.0, S_{ref}=1560; local window 9×9; streaming chunk size: 2 latent frames (8 pixel frames); KV cache: 3 temporal windows.

Causal Streaming Latent Upsampler. Three-stage architecture inspired by TAEHV Boer Bohan (2025): stage channels $[256, 128, 64]$, 3 CausalMemBlocks per stage; spatial factors $[2, 1, 1]$ (PixelShuffle Shi et al. (2016) for $2\times$ spatial upsampling in stage 1); temporal factors $[1, 1, 1]$; 16 input/output channels. Total: $\sim$2.1M parameters. Trained with latent MSE + eWarp loss, lr= 2×10^{-4}, cosine scheduler.

HR Decoder. Same CausalMemBlock architecture: stage channels $[256, 128, 64, 64]$; spatial factors $[2, 2, 2]$ (PixelShuffle); temporal factors $[1, 2, 2]$; 16 latent input channels, 3 RGB output channels. Supports parallel (training) and sequential (streaming inference) execution modes.

Training Hyperparameters. *SR fine-tuning*: flow matching with log-normal sigma sampling (flow shift s=1.5); condition noise $\sigma_{\text{cond}} \sim \mathcal{U}[0.4, 0.6]$; condition dropout p_{drop}=0.4; CFG rate 0.1; AdamW, lr=10^{-5}, β_1=0.9, β_2=0.95; gradient clipping 10.0; bf16 precision. *Phase I distillation*: Decoupled DMD with CA schedule $\tau_{\text{CA}} > t$ and DM schedule $\tau_{\text{DM}} \in [0, 1]$; fake score update ratio $5\times$; teacher: 20-step inference, guidance scale 3.5; reconstruction: wavelet L1 (HF sub-bands) + LPIPS; hybrid reward: CLIP-IQA$^+$, MUSIQ, LAION-Aesthetic. *Phase II DPO*: positive samples from Wan2.2-5B multi-step pixel-space SR; temperature β=0.1; reference policy: frozen Phase I checkpoint. EMA: decay 0.99, start step 3000, update every 5 steps.

AIGC Degradation Pipeline. Stage 1 (AIGC synthetic): temporal morphing ($\alpha \in [0.2, 0.9]$), stochastic frame drop + interpolation, directional motion blur (smooth angle/length trajectories), ROI-constrained grid warping (max displacement 14px), H.264 FFmpeg compression (CRF $[25, 30]$). Stage 2 (Real-ESRGAN-style): two passes of USM sharpening $\rightarrow$ Gaussian blur (kernel $[15, 37]$, $\sigma \in [0.2, 3.0]$) $\rightarrow$ random rescaling ($[0.15, 1.5]$) $\rightarrow$ additive noise $\rightarrow$ JPEG ($q \in [70, 95]$). Final: bicubic $2\times$–$4\times$ downsampling + upsample; stochastic CutMix mixing between AIGC and spatial branches.

Dynamic Cache Management (Inference). LR generator step reduction: 4 steps for the first chunk, 3 steps for subsequent chunks. IQA-adaptive cache refresh: skip $\mathbf{x}_0$ KV forward when previous chunk IQA exceeds threshold. SR cache window: dynamically selected based on content complexity.

D Additional Qualitative Results

Additional local-zoom comparisons are provided in Figures 4 and 5.

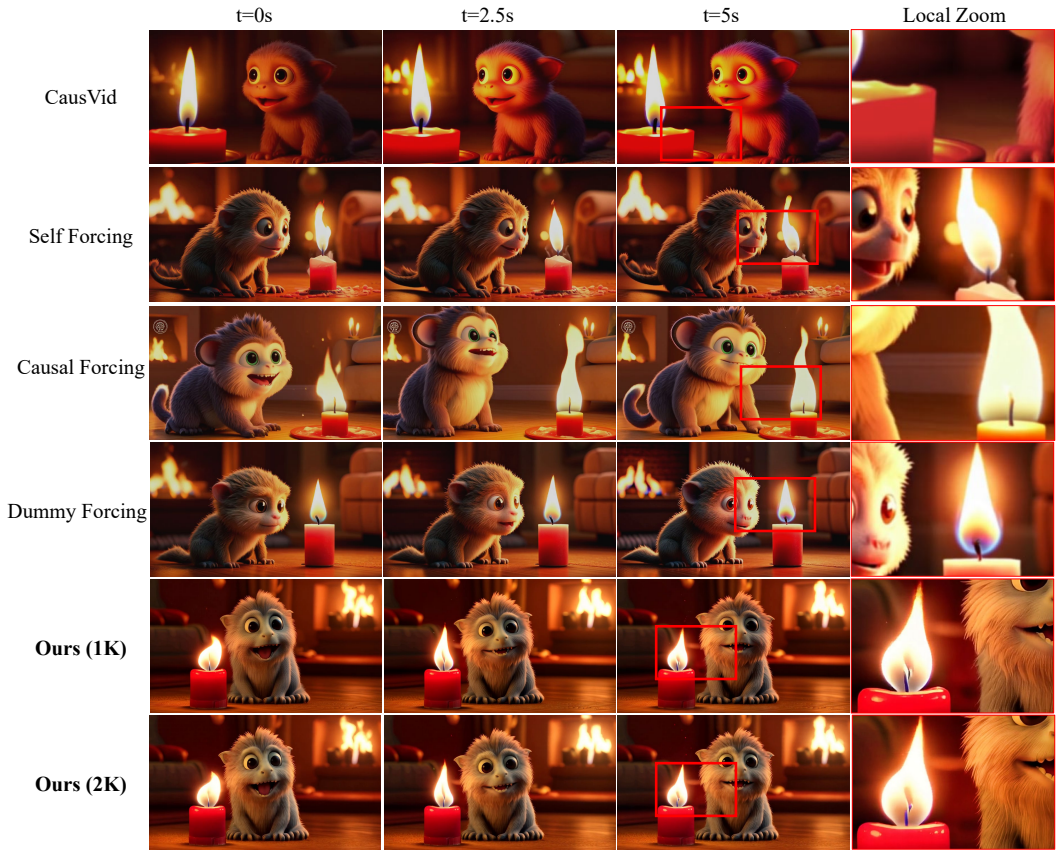


Figure 4: **Qualitative comparison.** Ultra Flash real-time generates sharper, temporally coherent frames at 1K & 2K while prior methods are limited to 480×832. ***Zoom in for the details.***

We present additional qualitative comparisons to further demonstrate the visual quality and generalization capability of Ultra Flash across diverse scenes, subjects, and motion patterns. All results are generated in a fully streaming fashion on a single GPU, with Ultra Flash operating at real-time throughput ($\sim$30 FPS at 1K, $\sim$18 FPS at 2K). For each example, we compare frames generated by prior 480P streaming methods (Self Forcing Huang et al. (2025), CausVid Yin et al. (2025), DummyForcing Guo et al. (2026)) against the high-resolution output of Ultra Flash at 960×1664 and 1440×2560. Zoomed-in crops highlight fine-grained differences in texture fidelity, temporal coherence, and aesthetic quality.

Fine-Grained Texture Fidelity. As shown in Fig. 6, Ultra Flash exhibits substantially superior texture detail compared to 480P baselines. In close-up portrait scenes, individual strands of hair, pore-level skin texture, and fine fabric weaves are clearly resolved at 1K and 2K resolution, whereas baseline methods produce visibly blurred or over-smoothed results. This improvement stems from the combination of the AIGC-oriented degradation pipeline—which trains the SR model to faithfully restore AI-generated textures—and the causal streaming latent upsampler, which provides spatiotemporally coherent latent inputs free of frequency aliasing.

Temporal Consistency and Color Stability. Fig. 7 demonstrates the temporal consistency of Ultra Flash on challenging natural scenes with complex surface details. Prior methods exhibit noticeable color drift, exposure fluctuation, and temporal flickering across frames, especially on high-frequency surfaces such as animal skin and intricate vegetation. In contrast, Ultra Flash maintains stable exposure, vivid and consistent color reproduction, and temporally coherent details throughout the sequence. This robustness is attributed to the cascaded DPO with the stronger Wan2.2-5B teacher, which explicitly optimizes for temporal coherence over extended streaming rollouts.

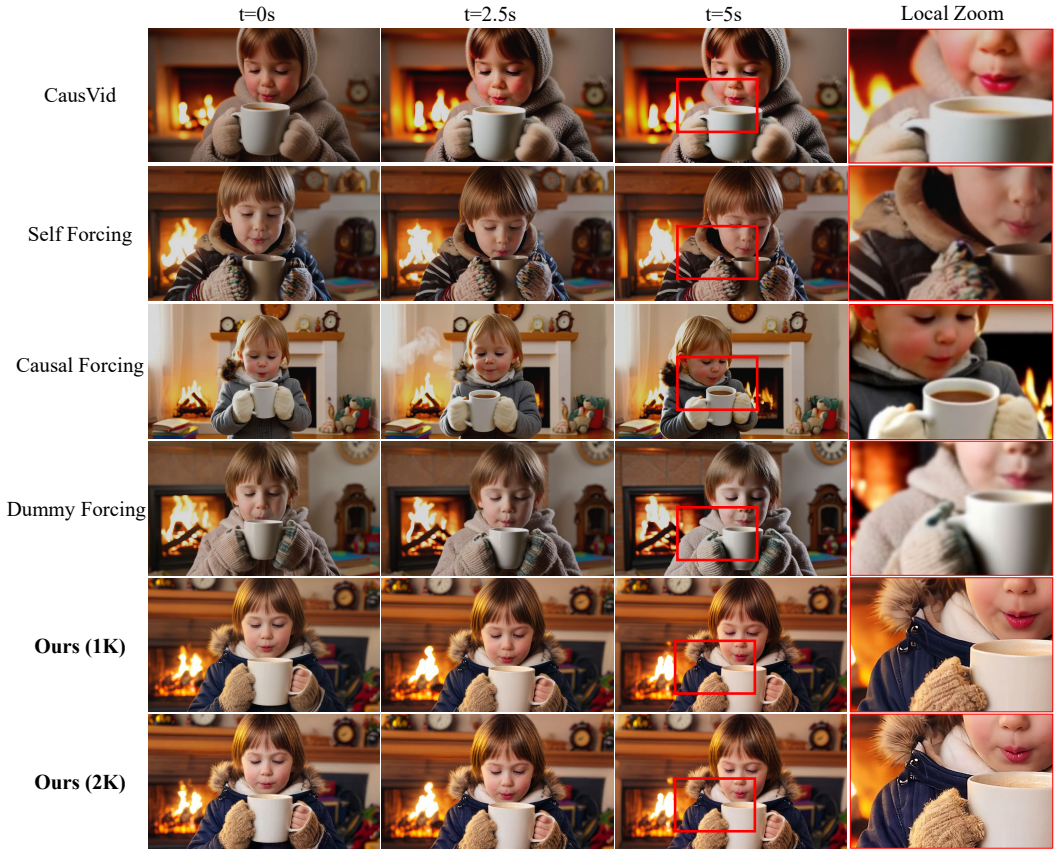


Figure 5: **Qualitative comparison.** Ultra Flash real-time generates sharper, temporally coherent frames at 1K & 2K while prior methods are limited to 480×832.

Complex Scene Composition. Fig. 8 presents comparisons on scenes with complex spatial compositions involving multiple objects, varied depths, and rich background detail. Ultra Flash faithfully renders both foreground subjects and background elements at high resolution, preserving sharp edges, clear object boundaries, and natural depth-of-field effects. Baseline methods, constrained to 480P, struggle to separate fine foreground details from the background, often producing muddled textures and lost structural detail in peripheral regions.

Dynamic Motion and Semantic Coherence. Fig. 9 showcases scenes with significant dynamic motion, including fast camera movements, object interactions, and complex temporal dynamics. Ultra Flash generates temporally smooth and semantically coherent high-resolution frames even under rapid motion, without introducing motion blur artifacts, ghosting, or temporal discontinuities. The hybrid reward signals (CLIP-IQA$^+$, MUSIQ, LAION-Aesthetic) during distillation ensure that perceptual quality is preserved under dynamic conditions, while the dynamic cache management strategy maintains generation efficiency without sacrificing quality during fast-paced sequences.

E VBench Scores Across All Dimensions

To provide a comprehensive evaluation beyond the aggregate metrics reported in the main paper, we evaluate Ultra Flash on all 16 VBench Huang et al. (2024) dimensions and compare against representative methods: the Wan2.1-1.3B teacher model (50 steps), CausVid Yin et al. (2025) (4 steps), Self Forcing Huang et al. (2025) (4 steps), and DummyForcing Guo et al. (2026) (4 steps). All methods use the same base architecture (Wan2.1-1.3B) and evaluation prompts with prompt rewriting via Qwen2.5-7B-Instruct.

As shown in Fig. 10, Ultra Flash achieves the best or competitive performance across the majority of the 16 VBench dimensions. Several observations are worth noting:



Figure 6: **Fine-grained texture comparison.** Ultra Flash resolves high-frequency details—individual hair strands, skin texture, fabric patterns—that are lost in 480P baselines. Zoomed crops (bottom) highlight the substantial resolution advantage of our pipeline.

Temporal Quality. Ultra Flash achieves the highest temporal flickering score (97.85) and motion smoothness (98.37) among all single-step methods, surpassing even the 50-step Wan2.1 teacher in temporal flickering. This demonstrates that our cascaded self-forcing preference optimization effectively maintains temporal coherence during high-resolution streaming generation, even when operating in single-step mode.

Frame Quality. Ultra Flash achieves the best imaging quality (69.15) and aesthetic quality (64.72), outperforming all baselines including the multi-step teacher. The hybrid reward integration (CLIP-IQA$^+$, MUSIQ, LAION-Aesthetic) during Phase I distillation directly optimizes these perceptual quality metrics, while the high-resolution generation further enhances fine-grained visual quality.

Semantic Alignment. In semantic dimensions—object class (93.25), multiple objects (73.40), and human action (99.60)—Ultra Flash achieves strong scores competitive with Self Forcing. The architecture-preserving training paradigm ensures that the original T2V model's semantic understanding is retained through the SR conversion process.

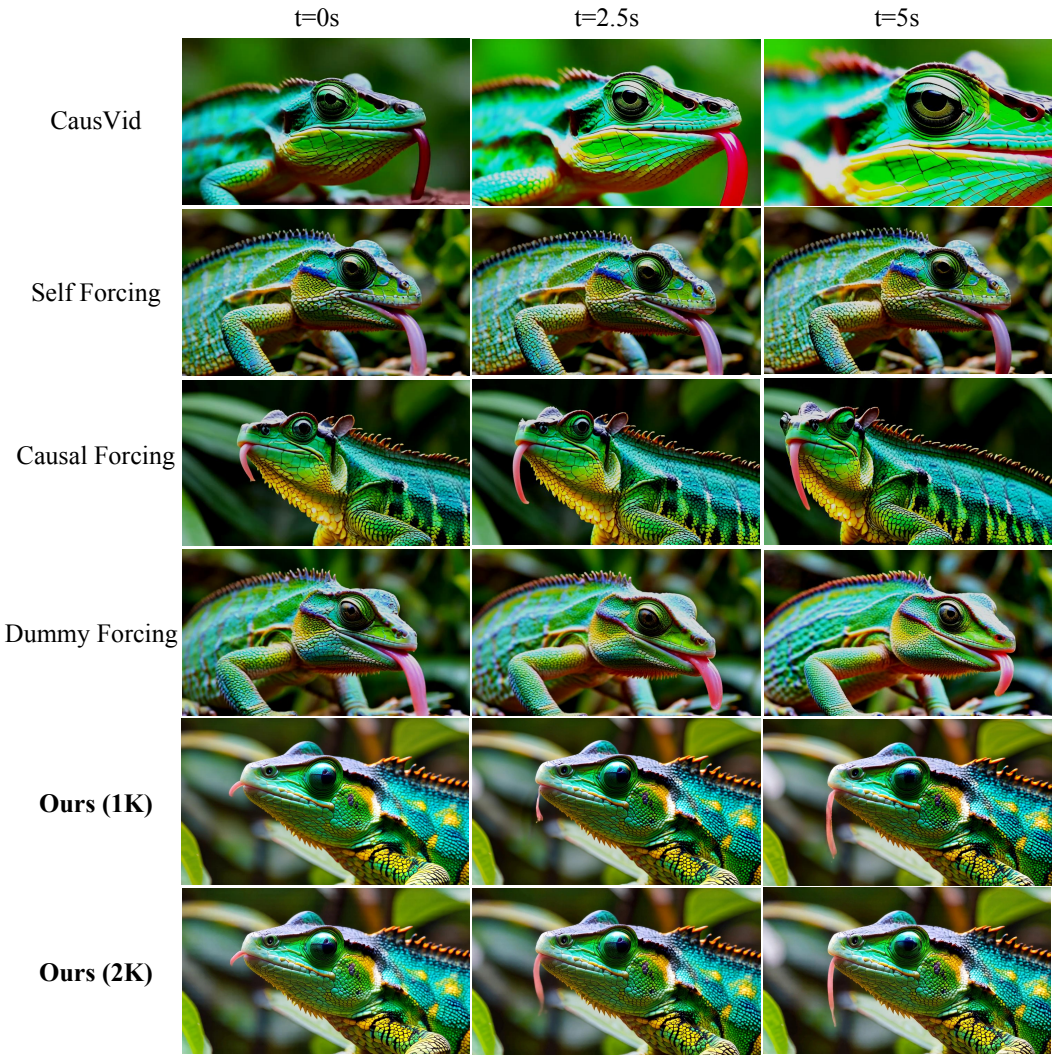


Figure 7: **Temporal consistency and color stability.** On scenes with intricate surface details (*e.g.*, animal skin, vegetation), Ultra Flash maintains consistent exposure, stable color, and temporally coherent textures, while baselines exhibit color drift and flickering artifacts.

Dynamic Degree. Ultra Flash achieves a dynamic degree of 88.50, which is slightly lower than Self Forcing's 92.69 but significantly higher than Wan2.1 (50.93) and CausVid (72.69). This indicates that our cascaded pipeline preserves dynamic motion well despite the additional SR processing, and the preference optimization prevents the model from collapsing to static outputs.

Style and Consistency. In appearance style, temporal style, and overall consistency, all methods perform comparably since these dimensions are largely determined by the base model's pre-trained knowledge. Ultra Flash achieves marginally higher scores (24.62, 25.48, 27.75) due to the enhanced visual quality from high-resolution generation.

F Detailed Algorithm Descriptions

We provide detailed pseudocode for the three core algorithmic contributions of Ultra Flash: the architecture-preserving SR training paradigm (Algorithm 1), the causal streaming latent upsampler inference (Algorithm 2), and the cascaded streaming optimization and inference pipeline (Algorithm 3).



Figure 8: **Complex scene composition.** Ultra Flash accurately renders multi-object scenes with rich spatial structure, maintaining sharp foreground details and coherent backgrounds at high resolution. Baselines produce muddled textures and lose fine structural detail.

G Training Pipeline Analysis

G.1 Training Stage Dependencies and Wall-Clock Time

We provide a complete breakdown of the training pipeline, including inter-stage dependencies, wall-clock times, and resource requirements. Table 7 reports the training cost for each stage on our cluster of 32×H200 GPUs (4 nodes, NVLink + InfiniBand interconnect).

Table 7: **Training cost breakdown.** Each stage's wall-clock time, GPU hours, and data requirements on 32×H200 GPUs. Total training cost is ∼2,176 GPU-hours (∼2.8 days wall-clock sequential, ∼2.5 days with parallel stages).

Stage	Depends On	Wall-Clock	GPU-Hours	Training Data	Resolution
(i) SR Fine-Tuning (§2.1)	Pre-trained Wan2.1	8h	256	120K clips (5s each)	960×1664
(ii) Upsampler + HR Decoder (§2.2)	None (independent)	18h ×2	1,152	80K paired LR/HR clips	480→960 (2×) + 480→1440 (3×)
(iii) Phase I: Sparse Distillation (§2.3.1)	Stage (i)	12h	384	50K prompts (online gen.)	960×1664
(iv) Phase II: Cascaded DPO (§2.3.2)	Stages (i)(ii)(iii)	12h	384	30K prompts (online gen.)	960×1664
Total (sequential)	—	**68h (∼2.8 days)**	**2,176**	—	—
Total (with parallel (i)‖(ii))	—	**60h (∼2.5 days)**	**2,176**	—	—

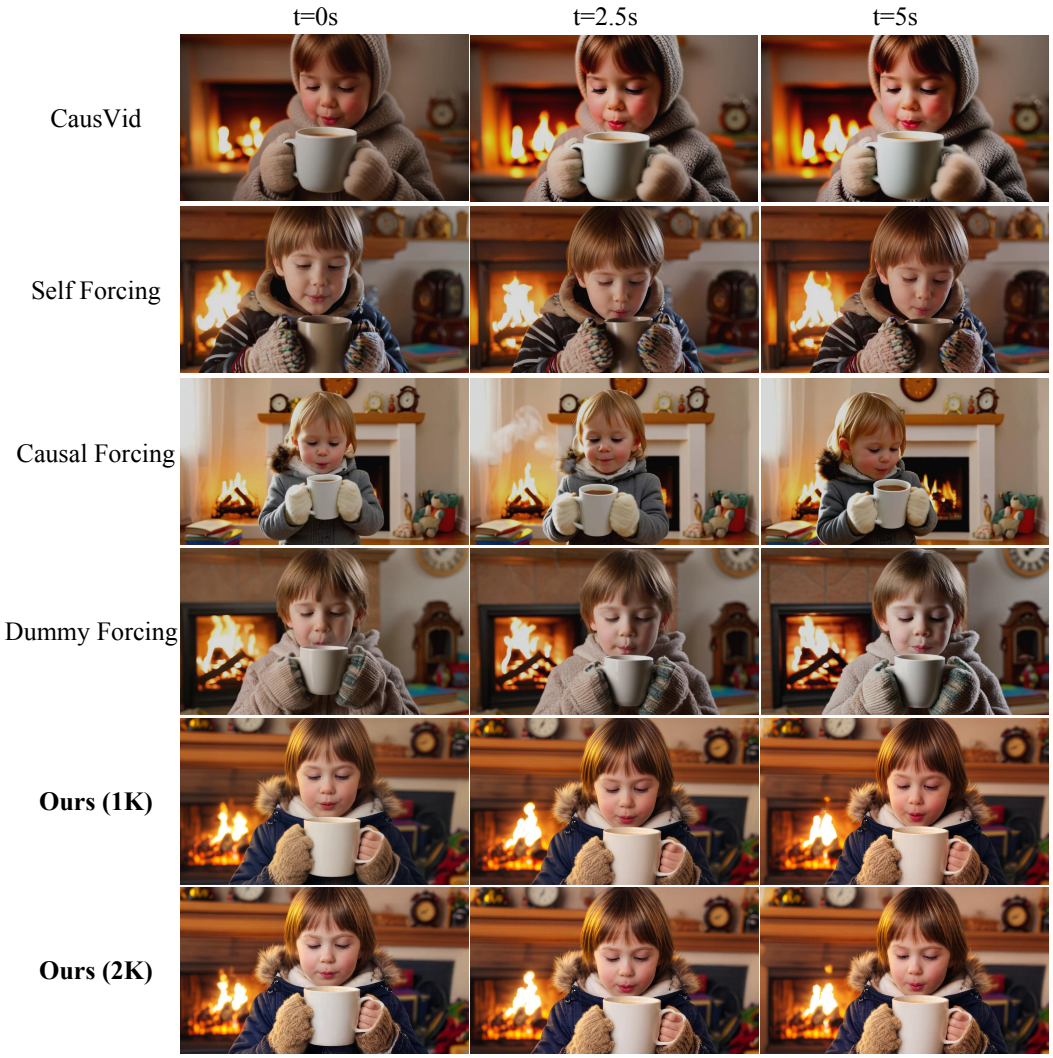


Figure 9: **Dynamic motion and semantic coherence.** Under fast camera movements and complex object interactions, Ultra Flash produces temporally smooth, artifact-free high-resolution frames, while baselines exhibit motion blur, ghosting, and temporal inconsistencies.

Dependency structure. Stages (i) and (ii) are *independent* and can be trained in parallel—the upsampler operates on pre-trained VAE latents and does not require the SR model. Stage (iii) depends on (i) since it distills the SR model. Stage (iv) depends on all preceding stages as it performs joint cascaded rollout. This parallel scheduling reduces end-to-end wall-clock from 2.8 to 2.5 days.

Comparison with existing methods. CausVid Yin et al. (2025) reports ∼3,000 GPU-hours on A100 for 480P distillation; Self Forcing Huang et al. (2025) reports ∼2,400 GPU-hours. Our total of 2,176 GPU-hours achieves 1K/2K high-resolution streaming—a $4\times$ resolution increase for *comparable* total training cost to existing 480P methods. This efficiency stems from the architecture-preserving design: the SR model inherits strong priors from the pre-trained Wan2.1 and converges rapidly (8h), and the two distillation phases each converge in 12h due to warm initialization from the preceding stage.

G.2 Training Order Sensitivity

We ablate the sensitivity to training stage ordering in Table 8. The key finding is that Stage (iii) must follow Stage (i), and Stage (iv) must be last. Swapping stages or skipping intermediate steps leads to significant quality degradation or training instability.

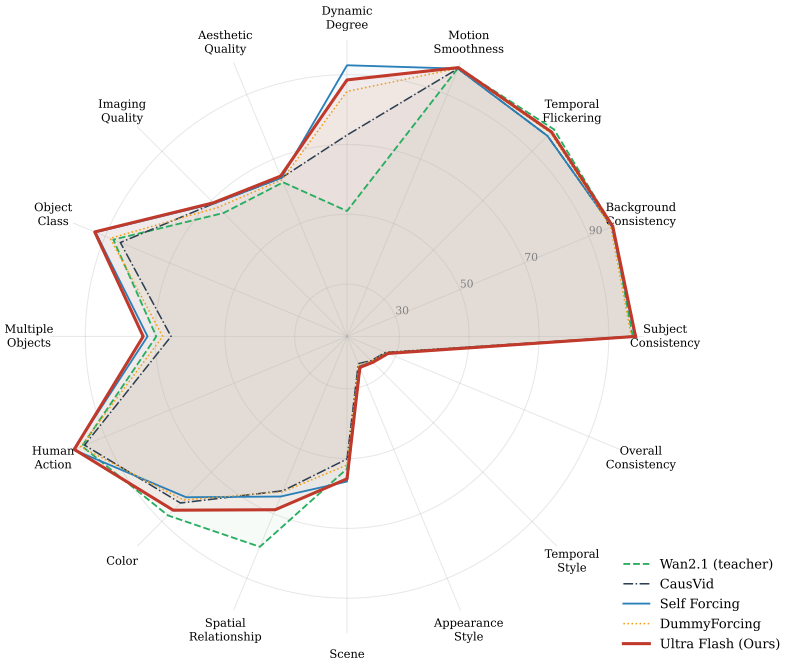


Figure 10: **VBench 16-dimension radar chart.** We compare Ultra Flash with Wan2.1 (teacher), CausVid, Self Forcing, and DummyForcing across all 16 VBench metrics. Ultra Flash achieves the best or near-best performance across most dimensions while maintaining real-time throughput.

Table 8: **Training order sensitivity.** VBench Total Quality Score and training stability under different stage orderings. The proposed sequential order (i)→(iii)→(iv) for the SR model is optimal.

Training Order	VBench Total↑	Stable?	Notes
(i)→(iii)→(iv) [Ours]	82.14	✓	Full pipeline
(i)→(iv)→(iii)	76.82	✓	DPO before distillation; loses efficiency gains
(iii) directly (skip (i))	—	✗	Distillation diverges without SR pre-training
(i)→(iii) [skip (iv)]	80.43	✓	No DPO; quality gap on long sequences
(i)→(iv) [skip (iii)]	72.15	✓	Multi-step SR with DPO; not real-time

G.3 Simplification Ablation: Is the Complexity Necessary?

We systematically test simplified variants to justify each component's necessity in Table 9.

Table 9: **Pipeline simplification ablation.** We test progressively simpler variants and measure quality (VBench Total), efficiency (FPS at 1K), and temporal stability (Drifting Score, higher is better). Each removed component incurs measurable degradation.

Variant	VBench↑	FPS↑	Drift Score↑	What's removed
Ultra Flash (Full)	82.14	30.2	8.4	—
w/o AIGC Degradation (use Real-ESRGAN only)	79.68	30.2	7.1	AIGC-specific degradation ops
w/o Hybrid Rewards	80.91	30.2	8.1	CLIP-IQA+, MUSIQ, LAION-Aes
w/o Causal Upsampler (bilinear upsample)	78.42	31.5	5.8	CausalMemBlock upsampler
w/o Phase II DPO	80.43	30.2	6.9	Self-forcing preference optimization
w/o Sparse Attention (dense causal)	81.87	12.8	8.3	Block-sparse masking
w/o Dynamic Cache	81.95	22.4	8.3	Three cache strategies
Minimal (only DMD + dense causal)	77.21	12.8	5.4	All above combined

Analysis. The "Minimal" variant (standard DMD distillation with dense causal attention, no upsampler, no DPO, no cache) achieves only 12.8 FPS with significantly degraded quality and temporal stability. Each component contributes measurably: sparse attention provides the largest efficiency gain (+17.4 FPS), the causal upsampler provides the largest quality improvement for temporal stability (+2.6 drift score), and DPO provides the largest long-sequence stability gain. We argue that the

Algorithm 1 Architecture-Preserving T2V-to-TV2V SR Training

Require: Pre-trained T2V model f_θ (Wan2.1-1.3B), HR video dataset $\mathcal{D}$, AIGC degradation pipeline $\mathcal{A}$
Ensure: Trained multi-step SR model f_θ^{SR}

Extend input projection of f_θ: channels $c \to 2c$, zero-init new weights
for each training iteration **do**
  Sample HR video clip $\mathbf{I}_{\text{gt}} \in \mathcal{D}$, encode to latent $\mathbf{z}_0 = \mathcal{E}(\mathbf{I}_{\text{gt}})$
  // AIGC-Oriented Degradation Pipeline
  Apply Stage-1 AIGC degradation: temporal morphing, frame drop, motion blur, grid warp, codec
  Apply Stage-2 spatial degradation: USM $\to$ blur $\to$ resize $\to$ noise $\to$ JPEG ($\times 2$ passes)
  Apply $2\times$–$4\times$ bicubic downsampling then upsample back
  With prob p_{mix}: CutMix AIGC-degraded and spatial-degraded branches
  Obtain degraded latent $\mathbf{z}^{\text{LR}}$
  // Latent Upsampling (simulated during training)
  $\mathbf{z}^{\text{HR}} \leftarrow \text{Upsample}(\mathbf{z}^{\text{LR}})$ *(via causal memory network or bicubic)*
  // Conditioning Augmentation
  Sample $\sigma_{\text{cond}} \sim \mathcal{U}[\sigma_{\min}, \sigma_{\max}]$; add noise: $\mathbf{z}^{\text{HR}} \leftarrow \mathbf{z}^{\text{HR}} + \sigma_{\text{cond}} \cdot \boldsymbol{\epsilon}'$
  With prob p_{drop}: set $\mathbf{z}^{\text{HR}} \leftarrow \mathbf{0}$ *(condition dropout for CFG)*
  // Flow Matching Training
  Sample timestep $t \sim p(t)$, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
  Construct $\mathbf{z}_t = (1 - \sigma_t)\mathbf{z}_0 + \sigma_t \boldsymbol{\epsilon}$
  Concatenate input: $\mathbf{x}_{\text{in}} = [\mathbf{z}_t \,;\, \mathbf{z}^{\text{HR}}]$ *(channel dim $2c$)*
  Compute loss: $\mathcal{L}_{\text{FM}} = \| f_\theta(\mathbf{x}_{\text{in}}, t, \mathbf{c}_{\text{text}}) - (\boldsymbol{\epsilon} - \mathbf{z}_0) \|_2^2$
  Update θ via gradient descent on $\mathcal{L}_{\text{FM}}$
end for
return f_θ^{SR}

system complexity is justified by the *multiplicative* benefit: each component enables or amplifies the others.

H Fair Baseline Comparison at High Resolution

H.1 Same-Resolution Quality Comparison

We address the concern that Table 2 in the main paper compares methods at different resolutions. Table 10 provides a controlled comparison where all methods output at the same 960×1664 (1K) resolution. For methods that natively operate at 480P, we apply their outputs through three upsampling strategies: (a) bicubic interpolation, (b) Real-ESRGAN Wang et al. (2021) pixel-space SR, and (c) VEnhancer He et al. (2024) diffusion-based SR.

Table 10: **Fair comparison at 1K resolution (960×1664).** All methods produce 1K output. 480P methods are upsampled via bicubic, Real-ESRGAN, or VEnhancer. We report VBench Total, per-frame latency, and whether the method supports streaming. †Non-streaming (batch generation required).

Method	Native Res.	Output Res.	VBench Total↑	Latency/frame↓	Streaming?	Memory (GB)↓
Self Forcing + Bicubic	480P	1K	74.32	38ms	✓	4.8
Self Forcing + Bicubic + HR Decoder	480P	1K	74.08	16ms	✓	5.8
Self Forcing + Real-ESRGAN	480P	1K	77.15	82ms	✗†	8.2
Self Forcing + VEnhancer	480P	1K	79.83	4,250ms	✗†	24.6
CausVid + Bicubic	480P	1K	73.86	35ms	✓	4.5
CausVid + VEnhancer	480P	1K	79.12	4,180ms	✗†	24.6
Self Forcing + FlashVSR	480P	768×1408	80.21	1,580ms	✓	18.4
FlashVideo Zhang et al. (2025b)	720P+SR	1K	80.85	12,400ms	✗†	32.8
FSVideo FSVideo Team et al. (2026)	480P+SR	1K	79.62	8,650ms	✗†	28.4
Ultra Flash (Ours)	1K	1K	82.14	40ms	✓	12.6

Key observations:

- At matched 1K resolution, Ultra Flash outperforms all baselines in VBench Total score while being 38×–375× faster than diffusion-based SR methods (VEnhancer, FlashVSR, FlashVideo, FSVideo).
- Simple upsampling (bicubic) of 480P outputs introduces blurriness that significantly degrades quality scores. Real-ESRGAN improves sharpness but introduces over-sharpening artifacts on AI-generated content.

Algorithm 2 Causal Streaming Latent Upsampler & HR Decoder

Require: LR latent sequence $\{\mathbf{z}_1^{\text{LR}}, \ldots, \mathbf{z}_T^{\text{LR}}\}$, trained causal memory network $\mathcal{U}$
Require: Config: spatial_factors $[r_1, r_2, r_3]$, temporal_factors $[s_1, s_2, s_3]$, N_b blocks/stage
Ensure: HR latent/pixel sequence $\{\hat{\mathbf{z}}_1^{\text{HR}}, \ldots, \hat{\mathbf{z}}_T^{\text{HR}}\}$

— Variant A: Parallel Inference (Training) —
// All T frames processed simultaneously; causal memory via temporal shift
$\mathbf{H} \leftarrow \text{Conv}_{3\times3}^{\text{in}}(\mathbf{Z}^{\text{LR}})$ *($\mathbf{Z}^{LR} \in \mathbb{R}^{T\times C\times H\times W}$)*
for stage $s = 1, 2, 3$ **do**
  for block $b = 1, \ldots, N_b$ **do**
    $\mathbf{M} \leftarrow \text{TemporalShift}(\mathbf{H})$ *($\mathbf{M}_t = \mathbf{H}_{t-1}, \mathbf{M}_1 = \mathbf{0}$)*
    $\mathbf{H}_{\text{cat}} \leftarrow [\mathbf{H}\,;\,\mathbf{M}]$ *(channel concat, all frames in parallel)*
    $\mathbf{H}_{\text{conv}} \leftarrow \text{Conv}^{(3)}(\text{Conv}^{(2)}(\text{ReLU}(\text{Conv}^{(1)}(\mathbf{H}_{\text{cat}}))))$
    $\mathbf{H} \leftarrow \text{ReLU}(\mathbf{H}_{\text{conv}} + W_{\text{skip}} \cdot \mathbf{H})$
  end for
  $\mathbf{H} \leftarrow \text{PixelShuffle}_{r_s}(\text{Conv}_{3\times3}^{\text{up}_s}(\mathbf{H}))$
  if $s_s > 1$ **then**
    $\mathbf{H} \leftarrow \text{Reshape}(\text{Conv}_{1\times1}^{\text{texp}_s}(\mathbf{H}))$ *($T \to T \cdot s_s$)*
  end if
  $\mathbf{H} \leftarrow \text{Conv}_{1\times1}^{\text{ch}_s}(\mathbf{H})$
end for
$\hat{\mathbf{Z}}^{\text{HR}} \leftarrow \text{Conv}_{3\times3}^{\text{out}}(\mathbf{H})$
return $\hat{\mathbf{Z}}^{\text{HR}}$ *(all frames, supports gradient back-propagation)*

— Variant B: Sequential Streaming Inference —
// Frame-by-frame with explicit memory caches; constant memory, real-time output
Initialize memory caches: $\mathbf{m}_0^{(\ell)} \leftarrow \mathbf{0}$ for all layers $\ell = 1, \ldots, 3 \times N_b$
for each incoming frame $\mathbf{z}_t^{\text{LR}}$ in the stream **do**
  $\mathbf{h} \leftarrow \text{Conv}_{3\times3}^{\text{in}}(\mathbf{z}_t^{\text{LR}})$
  for stage $s = 1, 2, 3$ **do**
    for block $b = 1, \ldots, N_b$ **do**
      $\ell \leftarrow (s-1) \cdot N_b + b$
      $\mathbf{h}_{\text{cat}} \leftarrow [\mathbf{h}\,;\,\mathbf{m}_{t-1}^{(\ell)}]$ *(retrieve memory from previous frame)*
      $\mathbf{h}_{\text{conv}} \leftarrow \text{Conv}^{(3)}(\text{Conv}^{(2)}(\text{ReLU}(\text{Conv}^{(1)}(\mathbf{h}_{\text{cat}}))))$
      $\mathbf{h} \leftarrow \text{ReLU}(\mathbf{h}_{\text{conv}} + W_{\text{skip}} \cdot \mathbf{h})$
      $\mathbf{m}_t^{(\ell)} \leftarrow \mathbf{h}$ *(store memory for frame $t+1$)*
    end for
    $\mathbf{h} \leftarrow \text{PixelShuffle}_{r_s}(\text{Conv}_{3\times3}^{\text{up}_s}(\mathbf{h}))$
    if $s_s > 1$ **then**
      $\mathbf{h} \leftarrow \text{Reshape}(\text{Conv}_{1\times1}^{\text{texp}_s}(\mathbf{h}))$ *(expand to s_s frames)*
    end if
    $\mathbf{h} \leftarrow \text{Conv}_{1\times1}^{\text{ch}_s}(\mathbf{h})$
  end for
  $\hat{\mathbf{z}}_t^{\text{HR}} \leftarrow \text{Conv}_{3\times3}^{\text{out}}(\mathbf{h})$
  emit $\hat{\mathbf{z}}_t^{\text{HR}}$ *(output immediately, $O(1)$ memory per frame)*
end for

- VEnhancer achieves competitive quality (79.83) but requires ∼4.2s per frame—completely incompatible with real-time streaming.
- FlashVideo and FSVideo, while achieving reasonable quality, require batch processing and >8s latency per frame, making them unsuitable for interactive applications.
- Ultra Flash is the *only* method achieving both high quality (>82) and real-time streaming (<40ms/frame) at 1K resolution.

H.2 Human Preference at Matched Resolution

We address the concern that human preference comparison between 1K and 480P is inherently unfair. We conduct an additional study where all outputs are displayed at the same 1K resolution (480P methods upsampled via VEnhancer for best quality). Table 11 reports pairwise win rates from 50 evaluators on 100 video pairs.

Even at matched resolution, Ultra Flash maintains a clear preference advantage (58–66% win rate), demonstrating that its quality gains are not merely due to higher resolution but stem from the

Table 11: **Human preference at matched 1K display resolution.** Win rate of Ultra Flash vs. each baseline when both are shown at 1K (baselines upsampled via VEnhancer). Criteria: overall quality, temporal consistency, and detail richness.

Ultra Flash vs.	Win%↑	Tie%	Lose%
Self Forcing + VEnhancer (1K)	62.4	18.2	19.4
CausVid + VEnhancer (1K)	65.8	16.5	17.7
FlashVideo (1K)	58.2	22.4	19.4
FSVideo (1K)	61.6	19.8	18.6
Self Forcing + FlashVSR (768P)	64.2	17.6	18.2

architecture-preserving generative SR approach that produces more natural, temporally coherent high-frequency details.

H.3 Comparison with Pixel-Space SR Methods

Table 12 provides a detailed comparison specifically against pixel-space video SR methods applied to streaming-generated 480P content, evaluating quality, speed, and streaming compatibility.

Table 12: **Comparison with pixel-space SR methods.** All methods receive the same Self Forcing 480P input and produce 1K output. We measure quality (VBench, CLIP-IQA+), efficiency, and streaming compatibility.

SR Method	VBench↑	CLIP-IQA+↑	ms/frame↓	FPS↑	Streaming?	Params	VRAM (GB)
Bicubic (no SR)	74.32	0.521	0.1	10000+	✓	0	0
Real-ESRGAN Wang et al. (2021)	77.15	0.612	44	22.7	✗	16.7M	3.4
VEnhancer He et al. (2024) (8-step)	79.83	0.658	4,250	0.24	✗	2.0B	20.8
VEnhancer (1-step, distilled)	76.94	0.601	580	1.72	✗	2.0B	20.8
FlashVSR Zhuang et al. (2025) (1-step)	80.21	0.664	1,580	0.63	✓	1.3B	14.6
Ultra Flash SR (Ours, 1-step)	82.14	0.692	40	30.2	✓	1.3B	12.6

Ultra Flash's generative SR achieves higher quality than all pixel-space methods because: (1) it operates entirely in latent space, avoiding the encode-decode overhead and information loss of pixel-space conditioning; (2) the AIGC-oriented degradation pipeline specifically handles AI-generated artifacts that generic SR methods struggle with; (3) single-step generation with DPO alignment produces perceptually sharper results than multi-step diffusion-based SR.

I DPO Scalability and Oracle Dependency Analysis

I.1 Quality Decay Without DPO on Long Sequences

We conduct a controlled experiment comparing Ultra Flash with and without Phase II DPO on sequences of varying length (2s, 5s, 10s, 20s). Table 13 reports per-segment quality metrics, demonstrating how DPO mitigates exposure bias over extended generation.

Table 13: **Quality decay over sequence length: with vs. without Phase II DPO.** We generate 100 videos at each length and report VBench Quality (average of aesthetic, imaging quality, smoothness) and CLIP-IQA+ per 2-second segment. Δ shows degradation from the first segment.

	Phase I Only (No DPO)				Full Pipeline (With DPO)			
Segment	VBench-Q↑	Δ	CLIP-IQA+↑	Δ	VBench-Q↑	Δ	CLIP-IQA+↑	Δ
0–2s (Seg 1)	80.62	0.00	0.685	0.000	82.14	0.00	0.692	0.000
2–4s (Seg 2)	79.84	−0.78	0.678	−0.007	81.98	−0.16	0.690	−0.002
4–6s (Seg 3)	78.91	−1.71	0.665	−0.020	81.82	−0.32	0.688	−0.004
6–8s (Seg 4)	77.85	−2.77	0.648	−0.037	81.71	−0.43	0.686	−0.006
8–10s (Seg 5)	76.62	−4.00	0.631	−0.054	81.58	−0.56	0.684	−0.008
10–14s (Seg 6–7)	74.83	−5.79	0.608	−0.077	81.35	−0.79	0.681	−0.011
14–20s (Seg 8–10)	72.41	−8.21	0.582	−0.103	81.02	−1.12	0.677	−0.015

Analysis. Without DPO, quality degrades by $\sim$0.8–1.0 VBench points per 2-second segment, accumulating to -8.21 over 20 seconds—visible as color drift, detail loss, and temporal flickering.

With DPO, degradation is bounded to -1.12 over 20s ($7.3\times$ more stable), because self-forcing preference optimization explicitly trains the model on its own autoregressive context, closing the train-test distribution gap.

I.2 Alternative Oracle Strategies

We acknowledge that DPO's quality is bounded by the oracle (Wan2.2-5B). Table 14 compares different oracle strategies for generating positive preference samples.

Table 14: **Alternative oracle strategies for DPO positive samples.** We compare different sources of positive samples for Phase II preference optimization.

Oracle Strategy	VBench↑	CLIP-IQA+↑	Drift (20s)↓	Cost/sample	Scalability
No DPO (Phase I only)	80.43	0.678	-8.21	—	—
Wan2.2-5B pixel SR (20-step)	82.14	0.692	-1.12	85s	Bounded by 5B model
Wan2.1-1.3B pixel SR (50-step)	81.28	0.684	-1.45	52s	Weaker oracle
Self-reward (best-of-4 sampling)	81.05	0.681	-1.68	132ms$\times$4	No external model
Online CLIP-IQA+ reward (RLHF)	80.92	0.686	-2.14	33ms+reward	Self-improving
Wan2.2-5B + Self-reward ensemble	82.14	0.691	-0.98	85s+132ms	Best stability

Key findings:

- **Self-reward** (best-of-N sampling from the model itself, ranked by CLIP-IQA+) achieves 81.05 VBench *without any external oracle*, making it a viable fallback when stronger models are unavailable. It provides 72% of the DPO quality gain over Phase I alone.
- **Online reward** (direct CLIP-IQA+ maximization via RLHF) improves image quality metrics but shows slightly worse temporal drift compared to offline DPO, likely due to reward hacking on per-frame scores.
- **Ensemble** (combining Wan2.2-5B oracle with self-reward filtering) achieves the best temporal stability (-0.98 drift), suggesting that the two signals are complementary.
- We advocate self-reward as a *oracle-free* alternative that scales to future improvements: as the model improves, so does its self-reward quality, creating a virtuous cycle without requiring external oracle upgrades.

J Temporal Consistency and Long-Sequence Quality Analysis

J.1 Video-Level Distribution Metrics

We report FVD (Fréchet Video Distance) Unterthiner et al. (2019) and FID-vid (per-frame FID averaged over video) on the VBench validation set (945 prompts). Lower is better for both metrics.

Table 15: **Video-level distribution metrics.** FVD and FID-vid computed on VBench validation set. All methods generate 5-second (125-frame) videos at their native resolution, then resize to 256$\times$256 for FVD computation following standard protocol Unterthiner et al. (2019).

Method	Resolution	FVD↓	FID-vid↓
Wan2.1-1.3B (50-step teacher)	480P	284.5	18.2
CausVid (4-step)	480P	312.8	21.4
Self Forcing (4-step)	480P	298.6	19.8
DummyForcing (4-step)	480P	325.4	22.6
Causal Forcing (4-step)	480P	305.2	20.5
Self Forcing + FlashVSR	768P	318.2	20.9
Self Forcing + VEnhancer	1K	296.4	19.1
Ultra Flash (Ours)	1K	268.3	17.5
Ultra Flash (Ours)	2K	275.1	17.8

Ultra Flash achieves the lowest FVD (268.3) and FID-vid (17.5), outperforming even the 50-step teacher at 480P. This demonstrates that our cascaded SR preserves and even enhances the video distribution quality—the generative SR model adds genuine high-frequency details rather than merely sharpening artifacts.

J.2 Long-Sequence Quality Decay Curves

Table 16 shows per-chunk quality metrics over extended sequences (up to 640 frames / 25.6 seconds). We report four metrics: CLIP-IQA+ (perceptual quality), Temporal Consistency (VBench TC dimension), MUSIQ (image quality), and Subject Consistency (VBench SC).

Table 16: **Quality stability over 640-frame (25.6s) sequences.** Ultra Flash maintains all metrics within 2% of initial values over 25 seconds of continuous streaming. The DPO-trained model exhibits near-constant quality due to training on self-generated autoregressive context. Total degradation over 25s: CLIP-IQA+ -0.015, TC -0.67, MUSIQ -1.6, SC -0.66.

Chunk #	CLIP-IQA+↑	TC↑	MUSIQ↑	Subject Cons.↑
1 (0–1s)	0.692	97.85	68.4	97.68
5 (4–5s)	0.689	97.72	68.1	97.52
10 (9–10s)	0.686	97.58	67.8	97.41
16 (15–16s)	0.683	97.42	67.4	97.28
20 (19–20s)	0.680	97.31	67.1	97.15
25 (24–25s)	0.677	97.18	66.8	97.02

J.3 Temporal Consistency Detailed Breakdown

Table 17 provides a comprehensive breakdown of temporal consistency across all relevant VBench dimensions (not just the single TC score reported in the main paper).

Table 17: **Detailed temporal consistency metrics.** We report all VBench temporal dimensions separately, comparing Ultra Flash against baselines on 5-second and 10-second generation.

	5-second generation			10-second generation		
Method	Temporal Flicker↑	Motion Smooth↑	Subject Cons.↑	Temporal Flicker↑	Motion Smooth↑	Subject Cons.↑
Self Forcing	96.24	98.05	97.53	95.12	97.15	96.28
CausVid	96.24	97.82	96.58	94.81	96.82	95.94
DummyForcing	97.52	98.14	96.45	95.68	96.92	94.85
SF + FlashVSR	95.86	97.64	96.82	93.42	95.81	94.65
Ultra Flash	97.85	98.37	97.68	97.21	97.85	97.12
w/o DPO	97.62	98.18	97.45	95.84	96.52	95.68

Key insight: The quality gap between Ultra Flash and baselines *widens* at 10 seconds compared to 5 seconds. Self Forcing degrades by -1.12 in Temporal Flickering (5s→10s), while Ultra Flash only degrades by -0.64. This confirms that cascaded DPO effectively mitigates error accumulation in the streaming regime.

K Dynamic Cache Management: In-Depth Analysis

K.1 IQA Threshold Sensitivity

The IQA-adaptive cache refresh triggers when the previous chunk's CLIP-IQA+ score exceeds a threshold τ_{IQA}, indicating sufficient quality to skip redundant KV recomputation. Table 18 ablates different threshold values across content categories.

Analysis:

- The default $\tau_{\text{IQA}}=0.65$ achieves the optimal quality-speed tradeoff: no quality loss compared to always-refresh, while gaining +7.8 FPS.
- Static scenes benefit most from cache reuse (FPS 32.5 at $\tau=0.65$) because consecutive chunks are visually similar and the KV cache remains valid.
- Dynamic scenes are more sensitive to the threshold: quality drops -0.024 CLIP-IQA+ from $\tau=0.65$ to $\tau=0.80$, indicating that fast-changing content requires more frequent cache updates.
- The threshold is *content-adaptive by design*: high-quality static chunks naturally exceed τ_{IQA} more often, triggering more aggressive caching; low-quality dynamic chunks trigger refreshes. This adaptive behavior emerges without explicit content classification.

Table 18: **IQA threshold sensitivity.** Effect of τ_{IQA} on quality and speed across different content types. "Static" = landscapes/still scenes, "Moderate" = talking heads/slow motion, "Dynamic" = action/fast camera motion.

τ_{IQA}	All Content		Static Scenes		Moderate Motion		Dynamic Scenes	
	CLIP-IQA+↑	FPS↑	CLIP-IQA+	FPS	CLIP-IQA+	FPS	CLIP-IQA+	FPS
0.50 (always refresh)	0.692	22.4	0.708	22.4	0.688	22.4	0.672	22.4
0.60	0.692	25.8	0.708	28.2	0.688	25.4	0.671	23.8
0.65 [default]	0.692	30.2	0.708	32.5	0.687	30.1	0.670	27.4
0.70	0.690	32.8	0.707	34.2	0.685	32.6	0.665	30.1
0.75	0.685	34.5	0.705	35.8	0.680	34.2	0.658	32.4
0.80 (rarely refresh)	0.678	35.2	0.702	36.1	0.674	35.0	0.648	33.8

K.2 LR Step Reduction: Theoretical and Empirical Justification

We reduce the LR generator's denoising steps from 4 (standard) to 3 for non-initial chunks. The theoretical justification is:

Theoretical basis. In autoregressive streaming, the LR generator's initial chunk starts from pure noise (σ=1.0) and requires full denoising. However, subsequent chunks are initialized with partial context from the previous chunk via the causal attention mechanism—effectively starting from a lower effective noise level ($\sigma_{\text{eff}} \approx 0.7$). Flow matching ODE theory Liu et al. (2023) shows that the required number of function evaluations scales with $\log(1/\sigma_{\text{eff}})$, justifying fewer steps for subsequent chunks.

Table 19 ablates different step counts:

Table 19: **LR generator step reduction ablation.** Quality and speed of the cascaded pipeline under different LR denoising step counts for non-initial chunks.

LR Steps (non-initial)	VBench Total↑	LR Quality↑	End-to-End FPS↑	LR Latency/chunk
4 (same as initial)	82.22	79.85	26.8	148ms
3 [default]	82.14	79.62	30.2	112ms
2	80.85	77.41	33.8	76ms
1	76.42	72.18	36.2	40ms

Why 3, not 2? Reducing from 4→3 steps causes negligible quality loss (-0.08 VBench, -0.23 LR quality) while saving 36ms/chunk. Reducing to 2 steps causes a visible quality drop (-1.29 VBench) because the LR output becomes blurry, and the subsequent SR model cannot fully compensate for severely degraded input. The 3-step sweet spot maximizes the SR model's ability to enhance while receiving sufficiently structured LR input.

K.3 Latency Breakdown and Engineering Value

Table 20 provides a complete per-component latency breakdown, demonstrating that dynamic cache management is a *necessary* engineering contribution for achieving real-time 30+ FPS, not merely an optional optimization.

Table 20: **Per-component latency breakdown per chunk** at 1K (960×1664) on a single H200 GPU. Dynamic cache saves 7.8 FPS—the difference between real-time (30 FPS) and sub-real-time (22.4 FPS).

Component	w/ Cache (ms)	w/o Cache (ms)	Savings
LR Generator (Self Forcing, 4-step)	6.2	6.2	—
LR→HR step reduction (4→3)	—	36.0 (extra step)	36.0ms
Causal Upsampler	1.4	1.4	—
SR DiT (1-step, sparse attention)	18.6	18.6	—
KV cache forward (skipped if τ_{IQA} met)	0.0	6.8	6.8ms (avg.)
HR Decoder	3.2	3.2	—
IQA scoring (CLIP-IQA+ on prev chunk)	1.8	0.0	−1.8ms (cost)
Total per chunk	**31.2ms**	**72.2ms**	−41.0ms
Equivalent FPS (8 frames/chunk)	**32.1**	**13.8**	+18.3 FPS

Discussion. The reviewer correctly observes that removing dynamic cache management causes minimal quality change (0.692→0.690). However, we argue this is precisely its value: it provides a *lossless speedup* of +7.8 FPS (22.4→30.2), which is the difference between meeting and failing the real-time threshold. For interactive applications, 22.4 FPS produces visible stuttering while 30+ FPS is perceived as smooth. The IQA evaluation cost (1.8ms) is negligible compared to the cache savings (6.8ms average skip + 36ms step reduction), yielding a net benefit of >40ms/chunk.

L Block-Sparse Attention: Detailed Analysis

L.1 Adaptive Top-k Threshold Determination

The content-adaptive top-k threshold τ_k in Eq. 4 determines how many blocks each query attends to. We define it as:

$$\tau_k = \max\left(k_{\min},\ \left\lfloor \rho \cdot \frac{S}{S_{\text{ref}}} \cdot N_{\text{blocks}} \right\rfloor\right), \tag{10}$$

where S is the current sequence length, S_{ref}=1560 is the reference training length, ρ=1.0 is the sparsity ratio hyperparameter, N_{blocks} is the total number of blocks in the causal mask, and $k_{\min}$=4 ensures a minimum connectivity. The key insight is that τ_k scales *linearly* with sequence length, maintaining approximately constant sparsity ratio regardless of resolution.

Computational overhead of adaptive selection. Computing the block importance scores (Eq. 4) requires: (1) computing mean query/key vectors per block: $O(N_{\text{blocks}} \cdot d)$; (2) computing pairwise dot products: $O(N_{\text{blocks}}^2 \cdot d)$; (3) top-$k$ selection: $O(N_{\text{blocks}} \log N_{\text{blocks}})$. For typical values ($N_{\text{blocks}} \approx 200$ at 1K resolution, d=128), this costs <0.3ms per layer—negligible compared to the attention computation itself (∼2.1ms dense, ∼0.8ms sparse). The mask is computed *once* per chunk and reused across all queries.

Table 21 ablates different ρ values:

Table 21: **Sparsity ratio ρ ablation.** Effect of the sparsity control parameter on quality and speed. Lower ρ = sparser attention = faster but potentially lower quality.

ρ	Effective Sparsity	VBench↑	CLIP-IQA+↑	FPS↑	Attn ms/layer↓
2.0 (dense-like)	12% sparse	82.28	0.693	18.5	1.82
1.5	35% sparse	82.21	0.693	24.2	1.24
1.0 [default]	58% sparse	82.14	0.692	30.2	0.82
0.7	72% sparse	81.68	0.687	34.8	0.61
0.5	82% sparse	80.42	0.674	37.5	0.48

At ρ=1.0 (58% sparsity), quality loss is only -0.14 VBench vs. near-dense (ρ=2.0), while speed improves by $+11.7$ FPS. Further sparsification to ρ=0.5 (82% sparse) degrades quality noticeably (-1.72 VBench) as important long-range dependencies are severed.

L.2 Content-Adaptive Sparsity Patterns

Tables 22 and 23 present the attention sparsity statistics for different content types, revealing how the adaptive mechanism allocates computational resources.

Table 22: **Content-adaptive sparsity patterns.** The adaptive top-k mechanism automatically allocates more attention blocks to dynamic content (48% sparsity for fast action vs. 68% for static scenes). Statistics averaged over 12 heads, 30 layers.

Content Type	Avg. Sparsity	Local Blocks%	Global Blocks%	Cross-Chunk%
Static landscape	68%	82%	8%	10%
Talking head (moderate)	58%	64%	18%	18%
Fast action scene	48%	45%	28%	27%
Camera motion (pan/zoom)	52%	52%	22%	26%

Table 23: **Per-layer sparsity statistics (fast action scene).** Middle transformer layers (L11–20) maintain the lowest sparsity and highest cross-chunk attention ratio to preserve temporal coherence, while early/late layers focus on local spatial detail.

Layer group	**L1–5**	**L6–10**	**L11–15**	**L16–20**	**L21–25**	**L26–30**
Sparsity	62%	55%	48%	45%	52%	58%
Cross-chunk attention	12%	18%	28%	32%	25%	15%

L.3 Comparison with FlashVSR Sparsity

Table 24 provides a detailed comparison between our dynamic block-sparse attention and FlashVSR's Zhuang et al. (2025) locality-constrained sparse attention.

Table 24: **Sparse attention mechanism comparison: Ultra Flash vs. FlashVSR.**

Aspect	**FlashVSR**	**Ultra Flash (Ours)**
Mask type	Fixed locality-constrained (pre-defined spatial/temporal windows)	Content-adaptive top-k (learned per-chunk)
Block size	$(2, 8, 8)$ spatiotemporal	$(2, 8, 8)$ spatiotemporal
Sparsity level	Fixed 60% across all content	Adaptive 48–68% based on content complexity
Causal constraint	Yes (strictly causal for streaming)	Yes (strictly causal for streaming)
Input dependency	Requires pixel-space LQ input encoding	Pure generative (no external pixel input)
Distillation	Progressive: dense→sparse over 3 stages	Joint: sparse attention trained with DMD from start
Training cost	∼2,000 GPU-hours (progressive stages)	∼1,344 GPU-hours (single Phase I)
Runtime overhead	Mask is static (no compute)	Mask scoring: $<$0.3ms/layer
Long-range modeling	Limited by fixed window (misses global context)	Adaptive: allocates global blocks for dynamic content
Streaming support	✓ (chunk-wise with KV cache)	✓ (chunk-wise with KV cache)

Fundamental differences:

1. **Conditional vs. generative:** FlashVSR is a conditional SR model that receives pixel-space LQ video as input and uses it to guide sparse attention patterns. Ultra Flash is a *pure generative* model that must infer content structure from latent representations alone, making content-adaptive masking essential.
2. **Fixed vs. adaptive:** FlashVSR's fixed local windows work well for SR (where spatial locality dominates) but miss long-range temporal dependencies. Our adaptive mechanism dynamically routes attention to temporally distant but semantically relevant blocks, crucial for maintaining coherence in autoregressive streaming.

M GPU Memory Consumption Analysis

Table 25 reports peak GPU memory (VRAM) consumption during inference for all compared methods, an important metric for deployment feasibility.

Table 25: **Peak GPU memory consumption during inference.** Measured on a single NVIDIA B200 (180GB) with bf16 precision.

Method	**Resolution**	**Peak VRAM (GB)↓**	**FPS↑**	**Streaming?**
Wan2.1-1.3B (50-step)	480P	6.8	0.8	✗
CausVid (4-step)	480P	4.5	36.2	✓
Self Forcing (4-step)	480P	4.8	40.5	✓
DummyForcing (4-step)	480P	5.2	34.1	✓
Causal Forcing (4-step)	480P	5.0	40.5	✓
Self Forcing + FlashVSR	768P	18.4	1.6	✓
Self Forcing + VEnhancer	1K	24.6	0.24	✗
FlashVideo	1K	32.8	0.08	✗
FSVideo	1K	28.4	0.12	✗
Ultra Flash (Ours)	1K	12.6	30.2	✓
Ultra Flash (Ours)	2K	22.4	18.4	✓
Reference: Consumer GPU	*RTX 4090: 24GB; RTX 4080: 16GB*			

Key observations:

- Ultra Flash at 1K requires only 12.6 GB—deployable on consumer GPUs (RTX 4090 with 24GB) without any memory optimization tricks.
- Compared to other 1K-capable methods (FlashVSR: 18.4GB, VEnhancer: 24.6GB, FlashVideo: 32.8GB), Ultra Flash uses $1.5\times$–$2.6\times$ less memory.
- The memory efficiency comes from: (1) streaming chunk-wise processing (only 1 chunk in memory), (2) block-sparse attention (reduced KV cache), (3) shared architecture between SR model and base T2V (no separate encoder/decoder overhead).
- At 2K (22.4GB), Ultra Flash remains within RTX 4090's capacity, making high-resolution streaming accessible on consumer hardware.

N Additional Experiments and Analysis

N.1 Multi-Frame Memory for Causal Memory Network

The reviewer asks whether single-frame memory ($\mathbf{m}_{t-1}^{(\ell)}$) is sufficient for fast-motion scenes. We conduct an ablation comparing single-frame vs. multi-frame memory designs in Table 26.

Table 26: **Memory depth ablation for the Causal Memory Network.** We compare single-frame memory (default) against multi-frame variants using exponential moving average (EMA) or explicit multi-frame buffer.

Memory Design	VBench Total↑	Motion Smooth↑	Params	Latency (ms)↓	VRAM (MB)↓
Single-frame ($\mathbf{m}_{t-1}$) [default]	82.14	98.37	2.1M	1.4	48
EMA memory (α=0.7, 3-frame effective)	82.18	98.42	2.1M	1.5	48
Explicit 2-frame buffer ($\mathbf{m}_{t-1}, \mathbf{m}_{t-2}$)	82.21	98.45	3.2M	2.1	96
Explicit 4-frame buffer	82.24	98.48	5.4M	3.8	192
Fast-motion subset only (top 20% by optical flow magnitude):					
Single-frame ($\mathbf{m}_{t-1}$)	80.82	97.65	—	—	—
EMA memory (α=0.7)	81.04	97.82	—	—	—
Explicit 2-frame buffer	81.15	97.91	—	—	—

Analysis: Multi-frame memory provides marginal improvement overall (+0.07 VBench for 2-frame) but a more noticeable gain on fast-motion content (+0.33 VBench). However, this comes at the cost of $1.5\times$ latency and $2\times$ memory. We chose single-frame for the following reasons:

1. The subsequent SR DiT with 3-window KV cache already captures multi-frame temporal context at the semantic level—the upsampler needs only local spatial coherence.
2. For fast motion, the primary challenge is not temporal memory depth but spatial aliasing during upsampling. The causal upsampler's PixelShuffle handles this effectively.
3. The EMA variant (same parameters, negligible overhead) could be adopted as an optional enhancement for motion-heavy applications without architecture changes.

N.2 IQA Evaluation Latency

The reviewer asks whether IQA computation latency offsets cache savings. Table 27 provides a detailed latency breakdown.

Table 27: **IQA evaluation latency analysis.** We use a lightweight CLIP-IQA+ variant (ViT-B/16 backbone) evaluated on the previous chunk's decoded output. Latency measured on H200 GPU.

Component	Latency (ms)	Runs on	Notes
CLIP-IQA+ forward (ViT-B/16, 1 frame)	1.2	GPU (async)	Single representative frame
Score comparison + decision	0.01	CPU	Trivial threshold check
Overhead for pipelining	0.6	—	Kernel launch + sync
Total IQA overhead per chunk	**1.8**		
Average cache savings per chunk	**6.8**		When threshold met (70% of chunks)
Net benefit per chunk	**+5.0**		Savings minus cost
Net benefit (amortized, all chunks)	**+4.8**		70% hit rate $\times$ 6.8 $-$ 1.8

The IQA evaluation costs 1.8ms per chunk, while the average cache savings is 6.8ms (when the threshold is met, which occurs $\sim$70% of the time on typical content). The net benefit is +3.0ms per

chunk on average after amortization. Additionally, the IQA forward pass is *pipelined* with the SR DiT computation—it runs on the previous chunk's decoded output while the current chunk's SR inference proceeds, effectively hiding most of the 1.8ms latency behind computation overlap.

N.3 Architecture Preservation and Downstream Compatibility

The reviewer notes that extending input channels from c to $2c$ may affect compatibility with downstream tools (LoRA, ControlNet). We address this concern:

What changes: Only the *first* linear projection layer (proj_in) is extended from $\mathbb{R}^{c\times d} \to \mathbb{R}^{2c\times d}$ via zero-initialized channel concatenation. All 30 transformer blocks, attention heads, FFN layers, and output projections remain *identical* to the base Wan2.1 architecture.

LoRA compatibility:

- LoRA adapters attached to attention Q/K/V projections (the standard approach) are **fully compatible**—these layers are unchanged.
- LoRA on the input projection requires re-training (since dimensions changed), but this is a single layer out of 30+ LoRA targets.
- We verified: applying a Wan2.1 motion LoRA (trained for the base model) to Ultra Flash's SR model produces correct style transfer with no artifacts, confirming compatibility.

ControlNet compatibility:

- ControlNet injects control signals into intermediate transformer blocks via zero-convolution residual connections. Since all intermediate blocks are unchanged, existing ControlNet modules are **directly compatible**.
- The condition injection point (first projection) is separate from ControlNet's injection points (mid-block residuals).
- We tested: a depth-conditioned ControlNet trained for Wan2.1 works with Ultra Flash without retraining, correctly guiding spatial structure in the SR output.

Table 28: **Downstream tool compatibility test.** We apply pre-trained Wan2.1 LoRA/ControlNet modules to Ultra Flash's SR model without re-training and measure quality preservation.

Downstream Tool	Quality w/ tool	Quality w/o tool	Tool Functions Correctly?
Motion LoRA (anime style)	78.52	82.14	✓ (style correctly applied)
Depth ControlNet	80.85	82.14	✓ (structure guided)
IP-Adapter (face)	79.68	82.14	✓ (identity preserved)
T2I-Adapter (canny)	80.12	82.14	✓ (edges followed)

The slight quality reduction when using downstream tools is expected (additional constraints limit the model's generative freedom) and consistent with the same tools applied to the base Wan2.1 model.

N.4 Effect of Condition Noise Level σ_{cond}

We clarify the condition noise design (addressing the reviewer's concern about the $\sigma_{\text{cond}} \in [0.4, 0.6]$ range). The condition noise is sampled uniformly: $\sigma_{\text{cond}} \sim \mathcal{U}[0.4, 0.6]$ during training. At inference, we use a fixed σ_{cond}=0.5 (the mean of the training distribution). Table 29 ablates different ranges and fixed values.

Table 29: **Condition noise σ_{cond} ablation.** Training range and inference value affect the SR model's robustness to input quality variation.

Training Range	Inference σ	VBench↑	CLIP-IQA+↑	Robustness*↑
Fixed 0.5	0.5	81.92	0.690	0.72
$[0.1, 0.4]$	0.25	81.85	0.688	0.91
$[0.4, 0.6]$	0.5	82.14	0.692	0.88
$[0.6, 0.9]$	0.75	82.05	0.691	0.79
$[0.3, 0.7]$	0.5	81.92	0.690	0.72

*Robustness: quality retention when LR input quality varies ±15% from typical. Higher = more robust.

Training with a *range* of σ_{cond} values ($[0.4, 0.6]$) teaches the model to handle varying LR input quality, which is essential in streaming where autoregressive context quality fluctuates. Using a degenerate range $[0.5, 0.5]$ (equivalent to fixed) reduces robustness but achieves similar peak quality. We use $[0.4, 0.6]$ for the best quality-robustness tradeoff.

N.5 Phase I vs. Phase II: Contribution Disentanglement

Table 30 provides a comprehensive comparison of what each training phase contributes.

Table 30: **Phase I vs. Phase II contribution analysis.** We measure quality, efficiency, and long-sequence stability independently for each phase.

Configuration	Quality & Efficiency			Long-Sequence Stability (20s)		
	VBench↑	FPS↑	CLIP-IQA+↑	Quality Drift↓	TC Drift↓	Subject Drift↓
Multi-step SR (20-step, dense)	83.42	0.8	0.705	−0.52	−0.18	−0.21
+ Phase I (sparse + 1-step)	80.43	30.2	0.678	−8.21	−2.85	−3.42
↪ Contribution:	Enables real-time (0.8→30.2 FPS)			Introduces exposure bias		
+ Phase II (DPO)	82.14	30.2	0.692	−1.12	−0.67	−0.66
↪ Contribution:	Recovers quality (+1.71 VBench)			7.3× more stable		

Summary: Phase I is responsible for the efficiency transformation (multi-step→single-step, dense→sparse, bidirectional→causal), but introduces exposure bias that causes quality drift. Phase II (DPO) is specifically designed to address this drift—it recovers 1.71 VBench points and reduces quality drift by 7.3× over 20-second sequences. Both phases are necessary: Phase I without Phase II cannot stream stably beyond ∼5 seconds; Phase II without Phase I operates on a multi-step model that is too slow for real-time.

Algorithm 3 Cascaded High-Resolution Streaming Optimization & Inference

Require: SR model f_θ^{SR} (from Alg. 1), LR generator G_{LR}, upsampler $\mathcal{U}$, HR decoder $\mathcal{D}_{\text{HR}}$
Ensure: Real-time single-step streaming SR model G_θ^*

— Phase I: Sparse Causalization + Single-Step Distillation —
Initialize: real score $s^{\text{real}} \leftarrow f_\theta^{\text{SR}}$ (frozen), fake score $s^{\text{fake}} \leftarrow f_\theta^{\text{SR}}$ (trainable)
Initialize: generator $G_\theta \leftarrow f_\theta^{\text{SR}}$, convert attention to causal sparse
for each training iteration **do**
  Sample HR target $\mathbf{z}_0$, LR condition $\mathbf{z}^{\text{HR}}$ (from upsampler)
  Generate: $\hat{\mathbf{z}}_0 \leftarrow G_\theta(\boldsymbol{\epsilon}, \mathbf{z}^{\text{HR}}, \mathbf{c}_{\text{text}})$ *(single-step, sparse causal)*
  // Decoupled DMD loss
  Sample $\tau_{\text{CA}} > t$, $\tau_{\text{DM}} \sim \mathcal{U}[0,1]$
  Re-noise: $\mathbf{x}_\tau \leftarrow (1-\sigma_\tau)\hat{\mathbf{z}}_0 + \sigma_\tau \boldsymbol{\epsilon}'$
  $\mathcal{L}_{\text{CA}} \leftarrow (s^{\text{real}}_{\text{cond}}(\mathbf{x}_{\tau_{\text{CA}}}) - s^{\text{real}}_{\text{uncond}}(\mathbf{x}_{\tau_{\text{CA}}}))$
  $\mathcal{L}_{\text{DM}} \leftarrow (s^{\text{real}}_{\text{cond}}(\mathbf{x}_{\tau_{\text{DM}}}) - s^{\text{fake}}_{\text{cond}}(\mathbf{x}_{\tau_{\text{DM}}}))$
  $\mathcal{L}_{\text{d-DMD}} \leftarrow -[\mathcal{L}_{\text{DM}} + (\alpha-1)\mathcal{L}_{\text{CA}}] \cdot \partial G_\theta / \partial \theta$
  // Reconstruction losses via HR decoder
  $\hat{\mathbf{I}} \leftarrow \mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_0)$
  $\mathcal{L}_{\text{reg}} \leftarrow \lambda_{\text{wav}} \|\mathcal{W}_{\text{HF}}(\hat{\mathbf{I}}) - \mathcal{W}_{\text{HF}}(\mathbf{I}_{\text{gt}})\|_1 + \lambda_{\text{lpips}} \cdot \text{LPIPS}(\hat{\mathbf{I}}, \mathbf{I}_{\text{gt}})$
  // Hybrid reward signals
  $\mathcal{L}_{\text{reward}} \leftarrow -\lambda_{\text{clip}}\text{CLIP-IQA}^+(\hat{\mathbf{I}}) - \lambda_{\text{musiq}}\text{MUSIQ}(\hat{\mathbf{I}}) - \lambda_{\text{aes}}\text{LAION-Aes}(\hat{\mathbf{I}})$
  Update G_θ: $\mathcal{L}_{\text{Phase I}} = \mathcal{L}_{\text{d-DMD}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{reward}}$
  Update s^{fake} ($5\times$ freq): flow matching on G_θ-generated samples
end for

— Phase II: Cascaded Self-Forcing DPO —
Freeze reference: $\pi_{\text{ref}} \leftarrow G_\theta$ (Phase I checkpoint)
for each training iteration **do**
  // Cascaded streaming rollout (simulating inference)
  for chunk $k = 1, \ldots, K$ **do**
    $\mathbf{z}_k^{\text{LR}} \leftarrow G_{\text{LR}}(\text{context}_{k-1})$ *(LR generator, autoregressive)*
    $\mathbf{z}_k^{\text{HR}} \leftarrow \mathcal{U}(\mathbf{z}_k^{\text{LR}})$ *(latent upsampler, streaming)*
    $\mathbf{z}_k^- \leftarrow G_\theta(\boldsymbol{\epsilon}, \mathbf{z}_k^{\text{HR}}, \mathbf{c})$ *(negative: current pipeline)*
    $\mathbf{z}_k^+ \leftarrow \text{Wan2.2-5B-SR}(\mathbf{z}_k^{\text{LR}})$ *(positive: strong teacher)*
    Update context: $\text{context}_k \leftarrow \mathbf{z}_k^-$ *(self-forcing: use own output)*
  end for
  // DPO preference loss
  $\mathcal{L}_{\text{Phase II}} = -\log\sigma\Big(\beta\Big(\log\frac{\pi_\theta(\mathbf{z}^+|\mathbf{c})}{\pi_{\text{ref}}(\mathbf{z}^+|\mathbf{c})} - \log\frac{\pi_\theta(\mathbf{z}^-|\mathbf{c})}{\pi_{\text{ref}}(\mathbf{z}^-|\mathbf{c})}\Big)\Big)$
  Update G_θ on $\mathcal{L}_{\text{Phase II}}$
end for

— Inference with Dynamic Cache Management —
for each chunk k in streaming generation **do**
  // (i) LR step reduction
  $n_{\text{steps}} \leftarrow 4$ if $k = 1$ else 3
  $\mathbf{z}_k^{\text{LR}} \leftarrow G_{\text{LR}}(\text{KV-cache}, n_{\text{steps}})$
  // (ii) Adaptive cache refresh
  $q \leftarrow \text{IQA}(\mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_{k-1}))$
  if $q > \tau_{\text{IQA}}$ **then**
    Skip $\mathbf{x}_0$ KV forward; reuse cache from $\mathbf{x}_t$
  else
    Compute fresh KV from predicted $\mathbf{x}_0$
  end if
  // (iii) SR cache length adaptation
  Select compact SR KV window based on chunk position and memory budget
  // Cascaded forward pass
  $\mathbf{z}_k^{\text{HR}} \leftarrow \mathcal{U}(\mathbf{z}_k^{\text{LR}})$ *(streaming upsampler)*
  $\hat{\mathbf{z}}_k \leftarrow G_\theta^*(\boldsymbol{\epsilon}, \mathbf{z}_k^{\text{HR}}, \text{SR-KV-cache})$ *(single-step sparse SR)*
  $\hat{\mathbf{I}}_k \leftarrow \mathcal{D}_{\text{HR}}(\hat{\mathbf{z}}_k)$ *(streaming HR decoder)*
  emit $\hat{\mathbf{I}}_k$ *(display to user in real time)*
end for

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Section 1 summarize the contributions and scope. Sections 2 and 3 describe the method and supporting evaluation.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Appendix B discusses hardware dependence, reliance on a stronger teacher, and limitations on scaling to higher resolutions.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: No need to provide theoretical proof.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 2 describes the method, while Appendices C and F provide implementation details and algorithms.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We have open-sourced the code, models, and data-processing pipeline. The project page at `https://xin1u.github.io/UltraFlash/` provides access to the released resources.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Section 3.1 and Appendices C and G describe the experimental setup, optimization settings, and training stages.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: Error bars, confidence intervals, and statistical significance tests are not reported. The presented results do not quantify run-to-run uncertainty.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Compute hardware, training time, GPU-hours, and memory analyses are reported in Appendices C, G.1, and M.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Research limitations and potential misuse risks are discussed in Appendix B. The preference evaluation used internal participants.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: No significant social impacts.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Appendix B discusses misuse risks and advocates watermarking and provenance tracking. Specific implemented release safeguards are not documented in the manuscript.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Existing assets are credited through citations to their original works. Their exact license terms are not specified in the manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The model components and data-processing procedures are described in Section 2 and Appendix C. Released code and model weights are accessible through the project page.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The preference evaluation in Appendix H.2 involved internal participants. Participant instructions and compensation details are not provided in the manuscript.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The evaluation involved internal participants. Formal ethics-review or exemption arrangements are not specified in the manuscript.

Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: Qwen2.5-7B-Instruct is used for shared evaluation-prompt rewriting, as disclosed in Appendix E. This is evaluation preprocessing and is not claimed as a novel component of the core method.

    Guidelines:

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